TREBALL FINAL DE GRAU

LECTURA REMOTA DE COMPTADORS ANALÒGICS

GRAU EN ENGINYERIA DE SISTEMES TIC

Autor: Raül Caldera Sànchez

Director: Pere Palà Schonwalder

Data: 7 de juliol de 2017

Localitat: Manresa

AGRAÏMENTS

Vull agraïr a tots els professors de l' EMIT de l' EPSEM, en especial al dr. Pere Palà Schonwalder, tutor d' aquest projecte, el fet d' haver fet possible la realització d' aquest projecte, i a totes les persones que m' han aconsellat o ajudat durant el seu transcurs.

RESUM DEL PROJECTE

Aquest projecte té el propòsit fer possible la lectura a distància de diferents comptadors analògics, per tal de tenir un millor control i seguiment del consum energètic, ja sigui d' una casa, d' una empresa, etc… i per facilitar la obtenció i posterior tractament de les dades que els comptadors ofereixen.

Malgrat que avui en dia existeixen els comptadors electrònics, els quals alguns ja envien les seves lectures a una base de dades remota, complint així el propòsit d' aquest projecte [ref. 75], pocs oferiexen encara aquest funcionalitat, i per altra banda encara és molt freqüent l' ús de comptadors analògics. Per això, en aquest projecte es proposa un sistema capaç de proporcionar la mateixa funcionalitat per a aquests comptadors.

Per tal de realitzar això, en aquest projecte s' ha construït un prototip del sistema, que consta d' una càmara per a prendre imatges del comptador, una Raspberry Pi que processa les imatges preses per la càmara, n' extreu les dades desitjades i les envia a un ordinador/servidor remot que guarda aquestes dades en una base de dades, ja preparades per al seu tractament, i també ofereix la possibilitat de poder-les passar a un format que es pugui llegir amb programes com el LibreOfficeCalc.

El projecte també consta del desenvolupament d' una montura que sosté el sistema físic (càmara, Raspberry Pi, comptador) i que conté elements per tal de maximitzar l' eficiència del sistema (miralls per a prendre imatges dels comptadors amb angles incilinats i que permeten reduïr l' espai que ocupa el sistema, un braç articulat per a moure la càmara, la possibilitat de incloure-hi díodes LED per a iluminar el comptador a la foscor…).

PROJECT ABSTRACT

The aim of this project is to offer the possibility to obtain remote readings from different meters, in order to achieve a better control and following of the energy consumption of a household, a company… and thus make the data obtention and treatment of energy consumption easier.

Despite nowadays electronic meters exist, some of which are capable of sending their reading to a remote database and therefore fulfilling the purpose of this project, few of them are able to do so, and moreover the use of analog meters is still quite frequent. Therefore, this project proposes a system capable of providing the same functionality for analog meters.

In order to do so, this project shows a prototype of the system, which consists of a camara to take pictures of the counter, a Raspberry Pi that gets such images, treats them to obtain the desired data, and sends the data to a remote computer/ server, which saves the data into a database and prepares it for posterior treatment, and also offers the possibility to transform such data to a readable format for programs like LibreOffice Calc.

The project also provides the development of a mount that supports the physical system (the camera, the Raspberry Pi, the energy meter), and that contains a series of elements to improve the overall efficiency of it, such as a mirror to take pictures with close angles to reduce the space needed for the system, an articulated arm that supports de camera, the possibility to include LEDs to illuminate the meter in dark spots…

Índex

Índex de figures

1. INTRODUCCIÓ

En aquest document s' exposarà com s' ha dut a terme aquest projecte, i s' explicaran pas per pas totes les parts que composen el sistema, el seu funcionament i les seves característiques. A part, també es detallarà quin és l' estat de l' art en l' àmbit del projecte, i es comentaran les similituds amb projectes anteriors.

A part d' això, es fa una estimació del cost dels components físics del prototip, i també compta amb un plec de condicions d' ús del sistema orientat a l' usuari per tal que pugui utilitzar el sistema de manera òptima. Finalment, el document mostra un conjunt de testejos del sistema per a demostrar el seu funcionament, i les conclusions que s' han extret de dites proves.

1.1 Objectius del projecte

- Realitzar un prototip de sistema capaç de llegir i processar dades de diferents comptadors, enviar-les a una base de dades remota i preparar-les per a la seva lectura i tractament.

- Aconseguir un sistema eficient i que es pugui adaptar a diversos problemes del medi (foscor, espai…).

- Assolir la màxima independència del sistema, per tal que es requereixi la mínima intervenció humana per al seu correcte funcionament.

- Que el sistema sigui econòmic i fàcil d' instal·lar i utilitzar.

1.2 Finalitat del projecte

- Ajudar a tenir un millor control del consum energètic, d' aigua …, ja sigui en una residència, en una empresa o en qualsevol altre lloc que es requereixi.

- Facilitar i automatitzar la consulta de tot tipus de comptadors analògics que ofereixin diversos tipus de dades.

1.3 Abast del projecte

El projecte s' ha realitzat tenint en compte un seguit de passos (tasques) que conjuntament han tingut com a objectiu comú la realització del prototipus del sistema. Els passos que s' han seguit han sigut, de manera cronològica:

- Tasca 1. Presa de informació sobre processat de dígits, OCR [ref. 33], thresholding d' imatge...
- Tasca 2. Realització del codi de processat d' imatge del comptador de llum.
- Tasca 3. Presa d' informació de protocols de transmissió de dades i d' imatges, entre Raspberry Pi [ref. 8] i Host. Estudi i implementació del sistema de pas de missatges amb protocol MQTT [ref. 29] i bases de dades.
- Tasca 4. Realització del codi de processat d' imatge del comptador d' aigua. Presa d' informació sobre processat d' agulles, detecció d' angles, etc …
- Tasca 5. Realització del disseny 3D de la montura.
- Tasca 6. Assemblatge del sistema amb la montura i testeig del sistema.
- Tasca 7. Redacció del document.

El calendari complet de la realització del projecte es mostra a continuació [Figura 1].

Tasca	Inici	End	Durada (dies)
Tasca 1.	02/20/17	03/13/17	21
Tasca 2.	03/06/17	04/17/17	42
Tasca 3.	03/06/17	03/27/17	21
Tasca 4.	03/13/17	05/01/17	49
Tasca 5.	04/17/17	05/22/17	35
Tasca 6.	05/22/17	07/03/17	42
Tasca 7.	05/15/17	07/03/17	49

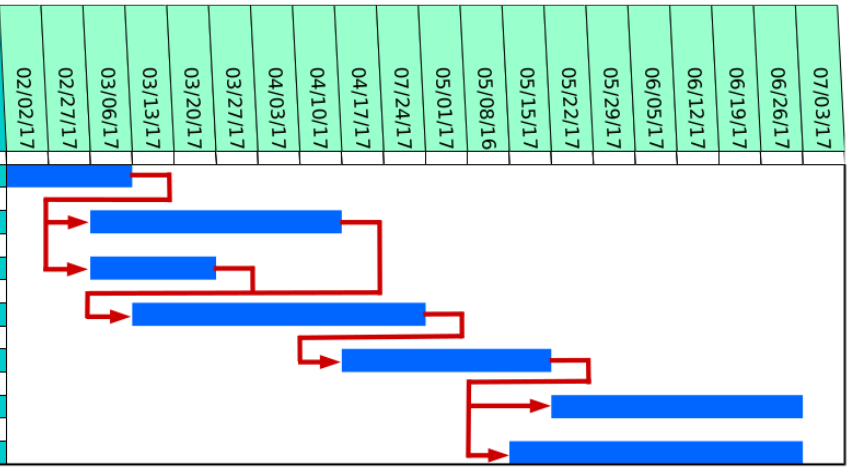


Figura 1: Diagrama de Gantt del projecte

2. ANTECEDENTS

2.1. Dades de partida

La base sobre la que es construeix el sistema d' aquest projecte són els comptadors analògics. Aquests comptadors s' utilitzen per a mesurar diversos tipus de consum energètic que es produeix a les cases, a les empreses, en els locals, oficines, etc... El consum que mesuren pot ser de diversos tipus, com per exemple energètic (Kwh), de l' aigua (m^3), del gas (m^3), etc …

Les dades sobre els diferents tipus de consum que marquen aquests comptadors són les que es volen tractar. A l' inici del desenvolupament d' aquest projecte tenim el comptador amb la informació que aquest ofereix. A partir d' aquí, es vol realitzar un sistema capaç de capturar aquestes dades i processar-les de manera que es puguin guardar en forma de text en una base de dades de manera periòdica, per tal de poder-les tractar i visualitzar-ne el consum que mostren al llarg del temps.

2.2. Estat de l' art

Actualment existeixen sistemes que realitzen el reconeixement de dígits, ja sigui de comptadors analògics, comptadors digitals, displays de 7 segments, etc… així com també existeixen sistemes de reconeixement d' agulles. Tot i així en l' actualitat la gran majoria dels sistemes semblen estar a nivell de prototipus o a nivell de idea, amb la qual cosa es pot dir que no hi ha una solució pràctica que sigui estàndard i es per això que cada sistema està construït de forma diferent.

Malgrat això, el processat d' imatge que es duu a terme en aquests sistemes és similar al que es duu a terme en aquest mateix projecte, ja que s' ha près com a influència un seguit de passos de processat bastant comú. Com a demostració, podem veure un resum gràfic del processat d' imatge que segueixen alguns projectes, com el de l' article *IoT Based Data Processing for Automated Industrial Meter Reader using Raspberry Pi* [ref. 1][Figura 2], el de l' article *Optical Character Recognition System for Seven Segment Display Images of Measuring Instruments* [ref. 2][Figura 3], i finalment el de l' article *A Low cost Data Acquisition System From Digital Display Instruments Employing Image Processing Technique* [ref. 3][Figura 4], els quals en tots s' executen processats d' imatge que inclouen reconeixement de dígits o similars, i tal com s' ha mencionat es veurà en els següents apartats d' aquest document que els passos de processat que es realitzen en el nostre sistema són semblants.

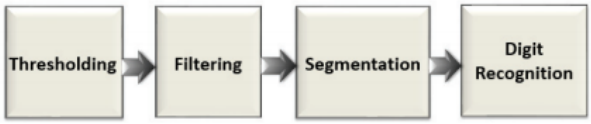


Figura 2: Passos de processat per a l' OCR, segons l' article IoT Based Data Processing for Automated Industrial Meter Reader using Raspberry Pi

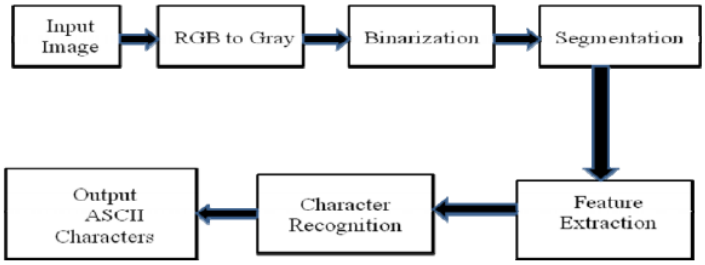


Figura 3: Passos de processat per a l' OCR, segons l' article Optical Character Recognition System for Seven Segment Display Images of Measuring Instruments

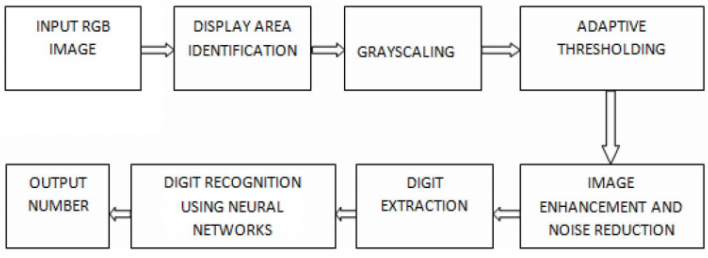


Figura 4: Passos de processat per a l' OCR, segons l' article A Low cost Data Acquisition System From Digital Display Instruments Employing Image Processing Technique

Tot i així les implementacions dels sistemes que es mostren en aquests articles són en general diferents. Els sistemes actuals solen centrar-se únicament en un sol tipus de processat, com per exemple el processat de dígits o bé el processat d' agulles, i rarament en ambdós àmbits com en aquest projecte.

Tot i que alguns d' aquests sistemes de forma similar al nostre sistema, actuant periòdicament per a prendre les dades i pujar-les a una base de dades o a Internet, no tots els sistemes actuen independentment, si no accionant un botó, com es pot veure en el sistema de l' article *IoT Based Data Processing for Automated Industrial Meter Reader using Raspberry Pi* [ref. 1] o a través d' una GUI (Graphical User Interface), com en el sistema que mostra l' article *Recognition of*

Numerals on Digital Meters in Dynamic Measuring [ref. 4] o també en el *A Low cost Data Acquisition System From Digital Display Instruments Employing Image Processing Technique* [ref. 3].

Pel què fa als aspectes de programació del processat d' imatge, aquest projecte s' ha influenciat notòriament en els passos que realitza el projecte *OpenCV practice: OCR for the electricity meter* [ref. 16] de Martin Kompf, en tot allò relacionat amb la detecció de dígits. D' altra banda, per a la detecció d' agulles, s' ha observat el mètode que es segueix en el projecte *Oil Level Reader* [ref. 71], de Chris Savina, així com també el del projecte *A-D converter the hard (but cheap!) way* [ref. 72], de Kenn Sebesta.

A l' hora de dissenyar la montura física del sistema, s' ha tingut en compte el disseny que es mostra en el projecte *Instrument Digitizer using Computer Vision* [ref. 73], realitzat per Edgar Almeida i Ivan Vodopiviz.

L' aspecte novedós que aporta aquest projecte és la possibilitat de realitzar lectures de diferents comptadors en funció de les necessitats de l' usuari, així com la realització de diferents tipus de processat en un sol sistema, ja que aquest és capaç de processar dígits i agulles a la vegada. Cada comptador requereix un cert processat, el qual és meticulós i s' ha de precisar amb detall, i per tant la intenció d' aquest sistema és adaptar-se a les necessitats de cada comptador.

2.3. Coneixements necessaris per a la realització del projecte

Per a realitzar i comprendre aquest projecte, es requereixen un conjunt de coneixements previs, i alguns coneixements que s' han adquirit durant la seva realització. Aquests són:

- Coneixements de processat d' imatge: detecció de dígits en una imatge, detecció d' agulles, càlcul d' angles, filtrat de contorns [ref. 32], detecció de colors…
- Coneixement bàsic de bases de dades i el seu tractament.
- Coneixements de disseny 3D.
- Coneixements de protocols de comunicacions, nocions de protocols i estructura d' Internet.
- Coneixements dels llenguatges de programació Python [ref. 9] i OpenSCAD [ref. 20].

3. EL SISTEMA

A continucació s' explicarà en profunditat el prototip realitzat. Es mostraran els components que formen el sistema, quines funcions realitzen i com interacccionen entre ells. En resum, es mostrarà què fa el sistema i com ho fa.

3.1. Descripció tècnica del sistema

El sistema [Figura 5] està composat d' una sèrie de components de caràcter físic i informàtic que units compleixen el propòsit de transmetre la informació dels comptadors a distància fins a un Host remot.

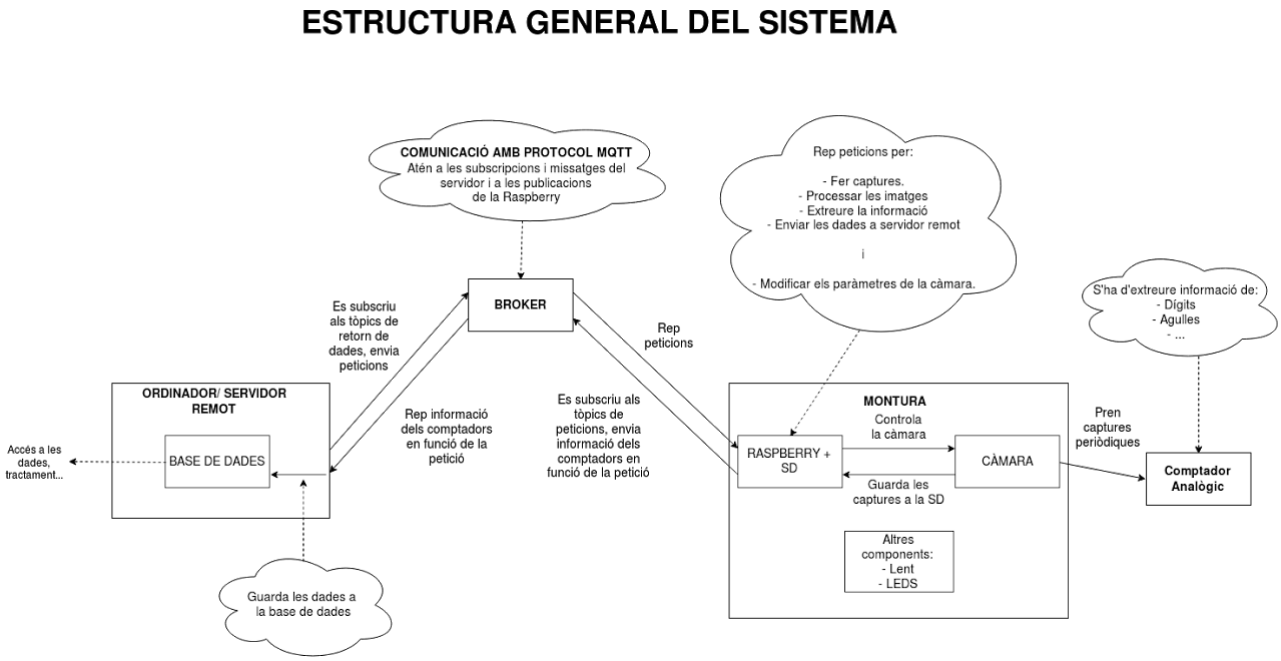


Figura 5: Esquema de funcionament general del sistema

En els següents subapartats s' entra en més detall en el paper de cada component.

3.1.1. La càmara

La càmara utilitzada en el prototipus és la PI NOIR CAMERA V2 [ref. 5][Figura 6]. És un mòdul que s' incorpora a la Raspberry Pi. Aquesta n' és la segona versió, i funciona amb tots els models de Raspberry Pi. Aquesta càmara té la característica que no conté un filtre d' infrarojos, amb la qual cosa permet veure-hi a la foscor amb una llum d' infrarojos. Aquesta era una possibilitat inicial i per això s' ha escollit aquesta càmara en comptes de la CAMERA MODULE V2 [ref. 6], tot i que la solució adoptada finalment ha sigut la d' adaptar la montura per tal que pugui incorporar-hi díodes LED [ref. 67] per a la iluminació.

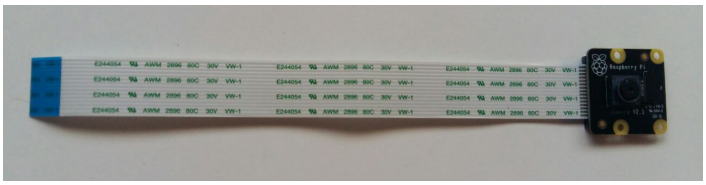


Figura 6: Pi Camera Noir V2.1

La càmara disposa de diferents configuracions que es poden modificar al gust de l' usuari, a través del seu mòdul de Python [ref. 7]. Per tal de facilitar-ne l' ús i el control, en el sistema és possible modificar aquests paràmetres (resolució [ref. 22], framerate [ref. 23], saturació [ref. 27], etc …), a través de la comunicació amb protocol MQTT [ref. 29] entre la Raspberry Pi i el Servidor/Client, tal i com s' ha mostrat anteriorment [Figura 5]. Més endavant s' explica amb més detall com s' utilitza aquesta funcionalitat.

3.1.2. La Raspberry Pi

Una Raspberry Pi es un mini-computador low-cost que està pensat per a usos informàtics de diferents tipus. Tot i que no inclou perifèrics, està dissenyada per tal que se n' hi puguin incorporar. No té la mateixa potència que un ordinador de taula o un portàtil, però pot realitzar-ne moltes de les funcions, amb l' avantatge d' oferir un baix consum i una alta portabilitat.

En el prototipus del sistema s' utilitza el model Raspberry Pi 3 Model B [ref. 8][Figura 7], ja que aquest incorpora connexió Wi- Fi (802.11n Wireless LAN), la qual cosa és necessària per a establir una connexió remota entre la part física del sistema (montura, càmara, Raspberry Pi, Comptador) i el Host.

Figura 7: Raspberry Pi 3

La Raspberry Pi [ref. 8] és el component més important del sistema, ja que abarca totes les seves facetes i està en contacte amb ambdós extrems d' aquest, i en realitza les funcions més essencials:

- Atén a les peticions del client i actua en consequència (captura de comptadors, modificació de paràmetres de la càmara).

- S' encarrega del processat d' imatge. Processa les captures per extreure' n la informació (processat dels dígits, de les agulles, OCR [ref. 33]).

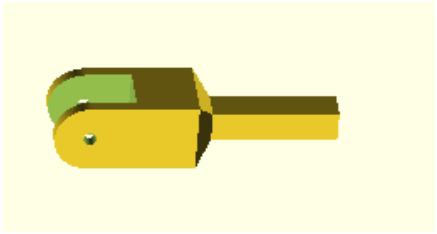
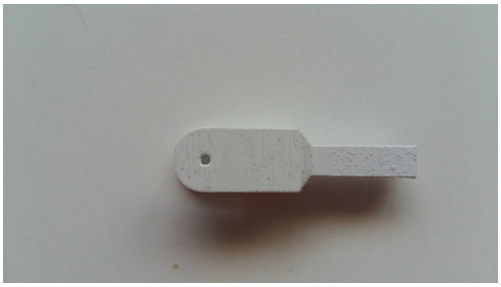
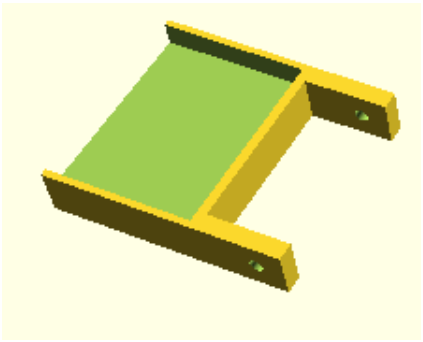
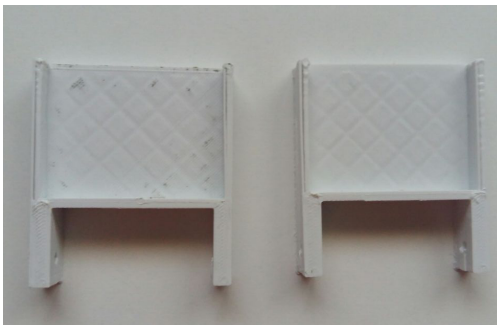
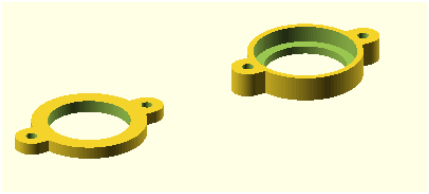
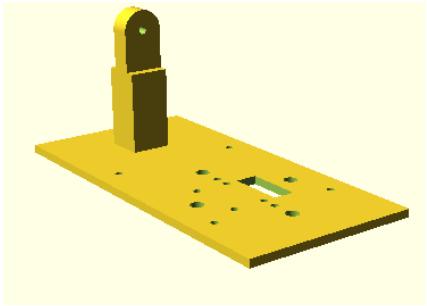
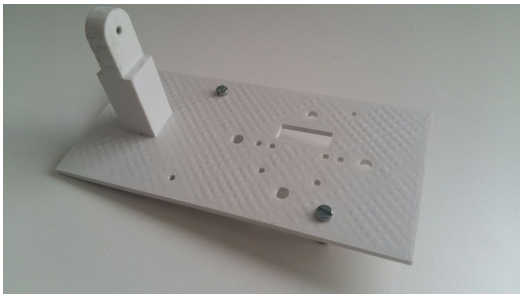
- Envia les dades de tornada al client, en funció de la petició sol·licitada.

3.1.3. La montura

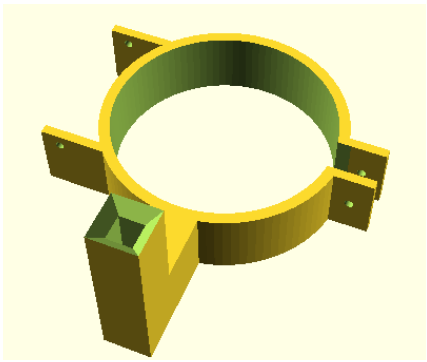
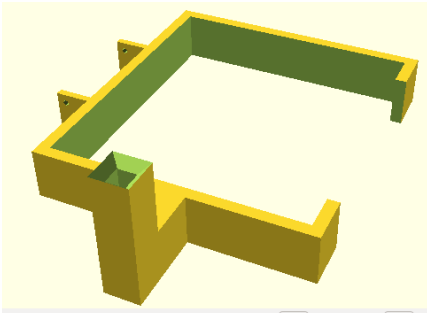
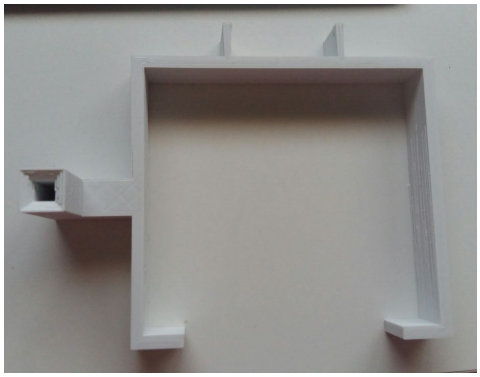
La montura s' encarrega de unir tots els elements físics del sistema. Es tracta d' un disseny imprimit amb una impresora 3D. L' estructura de la montura es pot organitzar en dues parts:

- Components aprofitables: aquests components estan dissenyats de manera que puguin ser aprofitables per a diferents tipus de comptadors. Per tant, si en algún moment es vol canviar de comptador per a prendre captures, aquests elements podran ésser els mateixos.

Els components aprofitables són el suport de la Raspberry Pi [ref. 8], el suport del mirall, el suport de la lent de la càmara [ref. 5] i el braç.

Descripció	Disseny 3D	Component imprès
Component del braç articulat		
Suports del mirall		
Suports de la lent de la càmara		
Suport de la Raspberry Pi		

- Components específics: són els components que suporten el propi comptador. Com que cada comptador té unes mides i formes diferents, han de tenir un suport propi.

Descripció	Disseny 3D	Component imprès
Suport central del comptador d' aigua		
Suport central del comptador de llum		

La montura s' ha dissenyat tenint en compte els següents aspectes:

- Que ocupi el mínim espai possible, per tal de potenciar la seva eficiència i fer el sistema el menys invasiu possible.

- Que pugui adaptar-se a l' entorn: en entorns tancats o molt petits, potser no és possible situar la càmara just davant del comptador. Per això la montura incorpora un braç articulat i un suport per a un mirall per tal de poder moure el suport de la Raspberry-càmara i minimitzar encara més l' espai que ocupa la montura. En aquests casos la càmara prendrà imatges del reflex del mirall que permetrà visualitzar la informació del comptador i processar-la amb èxit.

- Un altre aspecte que s' ha tingut en compte en aquest àmbit és la possiblitat de incloure leds per tal de iluminar comptadors en zones fosques, de nit … . La montura està pensada per a incorporar fins a 4 leds.

3.1.4. Codi necessari

En el sistema, hi ha diferents components els quals han de ser programats per tal de realitzar diverses tasques, com prendre imatges, fer el processat de la imatge, encarregar-se de la rebuda de peticions i transmissió de les dades… A continuació es mostra una llista dels llenguatges de programació i les llibreries que s' han utilitzat:

- Les llibreries més rellevants de Python [ref. 9] que s' han utilitzat són les següents:
    - **math**, per a realitzar càlculs per al processat de les imatges [ref. 10].
    - **time**, per a obtenir la data exacta de les captures [ref. 11].
    - **cv2**, la qual és la llibreria de Python de OpenCV [ref. 12]. Amb aquesta llibreria s' han utilitzat les funcions que ofereix per tal de fer tot el processat de les imatges.
    - **PIL**, una altra llibreria de tractament d' imatges, com a eina complementària de la llibreria `cv2` [ref. 13].
    - **pytesseract**, la llibreria de Python de tesseract [ref. 14], la qual s' utilitza per a fer el OCR [ref. 33].
    - **numpy**, per al tractament d' arrays de dades durant el processat de les imatges [ref. 15].
    - **picamera**, el mòdul de Python per a configurar la Pi Camera [ref. 7].
    - **paho-mqtt** (mosquitto), per a la comunicació de missatges entre la Raspberry Pi [ref. 8] i el Host via protocol MQTT [ref 17].
    - **sqlite3** per al tractament de la base de dades [ref. 18].
- Per a l' execució de processos temporitzats, s' utilitza **Cron** [ref. 19].
- Per al disseny 3D de la montura, s' ha utilitzat **OpenSCAD** [ref. 20].
- Per a la base de dades, s' ha utilitzat **SQLite** [ref. 21].

Amb l' ús de les llibreries que s' han mencionat anteriorment, s' han implementat els següents mòduls de Python i altres scripts, que conformen la part de software del sistema:

- `app.bd`: La base de dades.
- `images.sql`: fitxer que marca l' estructura de la base de dades i serveix per a inicialitzar-la.
- `db_operations.py`: mòdul de Python que ofereix funcions per a realitzar operacions sobre la base de dades.
- `exec1.sh`, `exec2.sh`, `exec3.sh`: scripts de shell per a executar diferents peticions sobre el protocol de pas de missatges entre el Host i la Raspberry Pi [ref. 8].
- `electricity_to_csv.sh`, `water_to_csv.sh`: scripts de shell per a executar operacions sobre la base de dades i passar el seu contingut a un fixer de format `csv`.
- `sub.py`: mòdul de Python que actua com a client (subscriptor) del protocol de pas de missatges a la màquina Host.
- `pub.py`: mòdul de Python que actua com a client (publicador) del protocol de pas de missatges a la Raspberry Pi [ref. 8].
- `camera_operations.py`: mòdul de Python que s' usa per a realitzar operacions sobre la càmara.
- `OCR_processing.py`: mòdul de Python que implementa les funcions que s' encarreguen de realitzar el processat d' imatge dels dígits del comptador de llum.
- `OCR_processing_water.py`: mòdul de Python que implementa les funcions que s' encarreguen de realitzar el processat d' imatge dels dígits del comptador d' aigua.
- `needle_procesing_water`: mòdul de Python que implementa les funcions que s' encarreguen de realitzar el processat d' imatge de les agulles del comptador d' aigua.
- `full_electricity_processing.py`: mòdul de Python que realitza tot el processat necessari del comptador de llum.

- `full_water_processing.py`: mòdul de Python que realitza tot el processat necessari del comptador de d' aigua.

- `electricity_support.scad`: script de OpenSCAD [ref. 20] on s' ha realitzat el disseny de la montura del comptador de llum.

- `water_support.scad`: script de OpenSCAD [ref. 20] on s' ha realitzat el disseny de la montura del comptador d' aigua.

- `rasp_support.scad`: script de OpenSCAD [ref. 20] on s' ha realitzat el disseny de la montura del suport de la Raspberry Pi i de la lent.

En els pròxims apartats s' entrarà amb més detall el paper que realitzen i el funcionament d' aquests fitxers.

3.2. Funcionament del sistema

En aquest apartat s' explicarà amb detall com funciona el sistema, com es realitza el processat de les imatges i com funciona el protocol de pas de missatges.

3.2.1. Dades que s' han de prendre dels comptadors

Per tal de fer el seguiment remot dels comptadors, s' han de prendre dos tipus de dades d' aquests. Per una banda, s' ha de prendre la informació sobre el consum que ofereix el comptador en un moment determinat, i per una altra s' ha de conèixer el moment en què s' ha près aquesta informació.

Cada comptador és diferent, i per tant s' hauran d' agafar unes dades determinades. En aquest projecte es prenen dades de dos comptadors, un comptador que mesura el consum elèctric i un altre que mesura el consum d' aigua. En el primer, s' han de prendre dades de 5 dígits que marquen el consum en kWh (KiloWatts-hora consumits) [Figura 8]. En el segon, s' han de prendre dades de 5 dígits i també de 3 dials amb 3 agulles [Figura 9], i que units marquen el consum d' aigua en m^3. Exactament, els primers quatre dígits marquen els milers, les centenes, les desenes i les unitats respectivament, el cinquè dígit marca les dècimes, el dial de la dreta marca les centèssimes, el dial del centre marca les milèssimes, i el dial de l' esquerra marca les dècimes de milèsima.

Per tant, el tractament de les imatges serà diferent en funció del comptador, tal i com es veurà a continuació.

Figura 8: Comptador de llum

Figura 9: Comptador d' aigua

3.2.2. Presa de les dades

Les dades que s' envien de la Raspberry al Host són les següents:

- El timestamp de la imatge, com a identificador.
- La imatge.
- Les dades de lectura del comptador.

La presa de de dades es realitza a través de la Pi Camera [ref. 6], la qual rep la ordre periòdicament de fer captures al comptador. El mòdul on s' implementa la captura i el setup de la càmara és el `camera_operations.py`.

Aquest mòdul implementa les funcions `camera_setup` i `shoot`. La primera funció crea l' objecte de camera i estableix els paràmetres de la càmara. Aquests paràmetres poden ser modificats a través del pas de missatges amb el protocol MQTT [ref. 29] amb el Host, tal i com es mostrarà més endavant. Tenen un valor numèric per defecte que s' estableix dins d' un rang de valors possibles. Els paràmetres en qüestió que s' ha establert que es poden modificar són:

- La resolució, per defecte 'HD' [ref. 22].
- El framerate, per defecte 15 [ref. 23].

- L' angle de rotació de la imatge, per defecte 0 [ref. 24].
- La nitidesa, per defecte 100 [ref. 25].
- La brillantor, per defecte 60 [ref. 26].
- La saturació, per defecte 100 [ref. 27].
- Girar la imatge, horitzontalment i verticalment [ref. 74].

La funció `shoot` és la que s' encarrega de realitzar la captura. La imatge es captura en forma de stream de bytes [ref. 28] ja que s' ha de transmetre al servidor via MQTT [ref. 29] en aquest format. Per al processat no hi ha problema amb aquest format ja que OpenCV [ref. 12] pot treballar amb imatges com a stream de bytes. A mesura que es capturen imatges, aquestes no es guarden a la SD, per tal de no acabar ocupant tota la memòria. La funció simplement retorna l' stream de bytes. Totes les imatges es sobreescriuen posteriorment en el fitxer `original.jpg`. La funció també retorna el timestamp que serà el nom identificador de la imatge. Tant l' stream de bytes com el timestamp seran enviats posteriorment, juntament amb l' string de dades corresponent a la lectura del comptador.

3.2.3. Tractament de les dades

Un cop presa la imatge, es passa a fer-se el processat per tal d' extreure' n les dades. Com s' ha esmentat anteriorment, per a cada comptador s' ha de fer un processat diferent, degut a que les dades s' han d' extreure de diferents maneres (dígits, agulles, diferents colors…).

A continuació s' explica com es realitza el processat de les imatges per al comptador de llum i per al comptador d' aigua.

3.2.3.1. Processat d' imatge del comptador de llum

La imatge a processar del comptador de llum és com la que s' ha mostrat anteriorment [Figura 8]. L' objectiu és reconèixer els dígits de la imatge que pertanyen al consum energètic i fer l' OCR d' aquests dígits [ref. 33]. Per tal de fer això es segueixen aquests passos:

- Es realitza un Canny edge detector a la imatge [ref. 30], per a detectar-ne els eixos i facilitar-ne el posterior processat.
- Com que la imatge pot estar una mica torçada, s' alinia rotant-la amb la inclinació necessària per tal de deixar-la recta. Això es fa amb el mètode de Houghlines [ref. 31].

- Un cop amb la imatge en format Canny i aliniada, es busquen els contorns de la imatge [ref. 32], és a dir, les formes que hi ha la imatge producte dels eixos trobats.

- Dins del conjunt de contorns trobats, hi ha d' haver els que corresponen als dígits del comptador. Per tant, de tots els contorns, s' han de filtrar aquells que pertanyin als dígits. Aquest fitrat consisteix primerament en agafar els contorns d' una certa alçada i amplada que puguin correspondre als contorns dels dígits, i després filtrar-los en funció de la seva posició en la imatge. Com que els dígits estaran gairebé en una mateixa posició Y (vertical) de la imatge, s' ha de veure quins dels contorns poden ser els dígits en funció d' aquesta posició Y. També han d' estar separats entre ells unes distàncies més o menys iguals en la seva posició X, ja que els dígits estan un al costat de l' altre.

- Un cop trobats els contorns filtrats, aplicant el criteri de les posicions, es fan retalls de la imatge original de forma recangular a la posició de la imatge on es creu que hi ha cada dígit. A cada imatge, que correspon a un sol dígit, se li aplica un thresholding per a passar-la en blanc i negre, per tal de facilitar el posterior OCR [ref. 33].

- Finalment, amb la imatge de cada dígit amb el thresholding aplicat, es fa l' OCR de cada dígit, un per un, passant el valor trobat a un string, preparat per a ser transmès.

Totes els passos descrits anteriorment es realitzen en les funcions del mòdul `OCR_processing.py`, i tot el processat es pot realitzar executant el mòdul `full_electricity_processing.py`, que corre totes les funcions i retorna l' string de dades dels dígits.

Les funcions de `OCR_processing.py` són les següents:

- `image_to_canny(img)`: La funció rep com a paràmetre l' stream de bytes [ref. 28] corresponent a la captura que ha fet la càmara, i guarda la imatge amb el nom `original.jpg`. A continuació, transforma la imatge en blanc i negre amb la funció `cv2.cvtColor` [ref. 34]. A la imatge en blanc i negre se li aplica llavors el Canny edge detector [ref. 30] amb la funció `cv2.Canny` [ref. 35]. Es guarda la imatge Canny amb el nom `electricity_meter_canny.jpg` [Figura 10]. La funció retorna la variable de la imatge Canny.

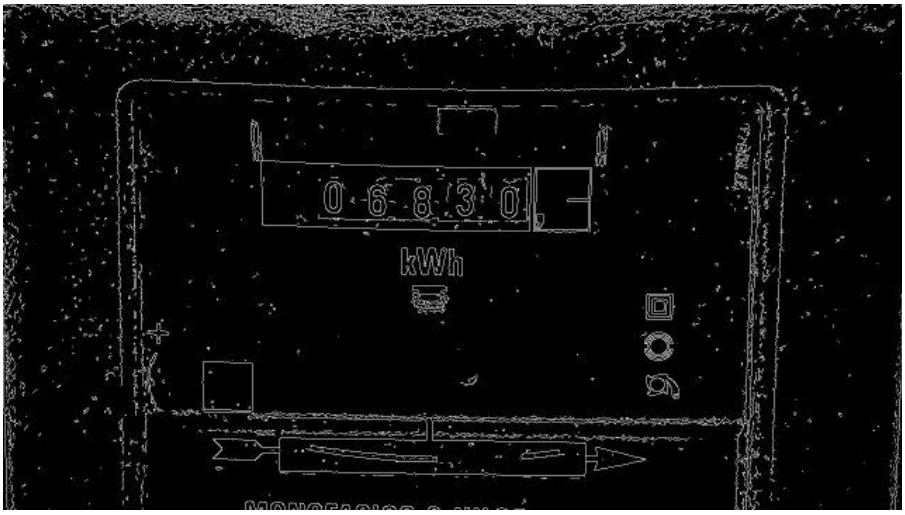


Figura 10: Imatge del comptador de llum en format Canny

- `houghlines_detection(img, canny_img)`: Els paràmetres de la funció són l' stream de bytes [ref. 28] de la imatge original i la imatge Canny que retorna la funció `image_to_canny`. Aquesta funció és la que s' encarrega d' aliniar horitzontalment la imatge. Això ho fa amb el mètode Houghlines [ref. 31], que s' utilitza per a detectar línies rectes en una imatge.

Amb aquest mètode implementat per la funció `cv2.HoughLines` [ref. 36] es detecten línies rectes que haurien de tenir un angle de desviació de 0 graus respecte l' eix horitzontal, però que en canvi estan una mica desviades, indicant així que la imatge està torta. Tenint en compte aquest angle de desviació, es rota la imatge en sentit contrari, amb ajuda de les funcions `cv2.getRotationMatrix2D` [ref. 37] i `cv2.warpAffine` [ref. 38]. El funcionament de `cv2.Houghlines` [ref. 36] s' explica amb més detall a l' apartat **3.2.3.2.**

Es guarda la imatge amb les línies trobades amb el nom `Houghlines.jpg` [Figura 11] i la imatge en format Canny orientada horitzontalment amb el nom `edges_aligned.jpg` [Figura 12].

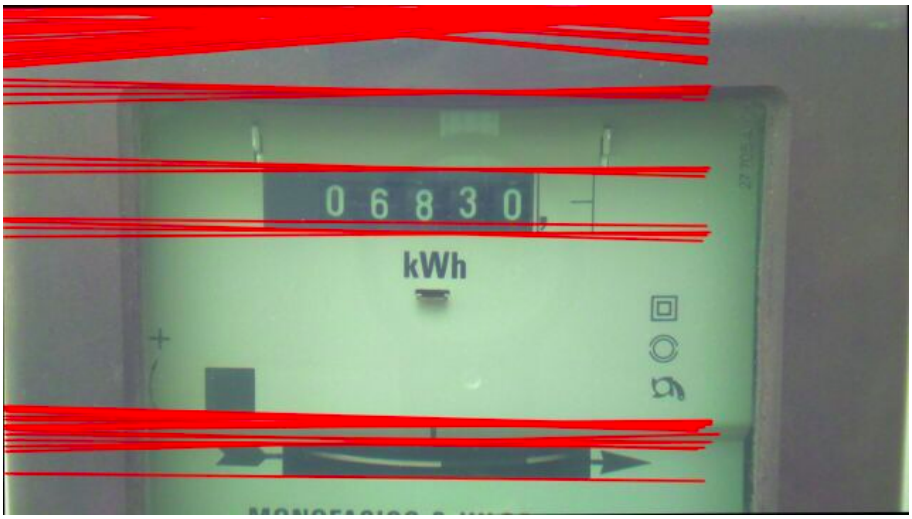


Figura 11: Imatge de les línies horitzontals trobades en el comptador de llum amb el mètode HoughLines

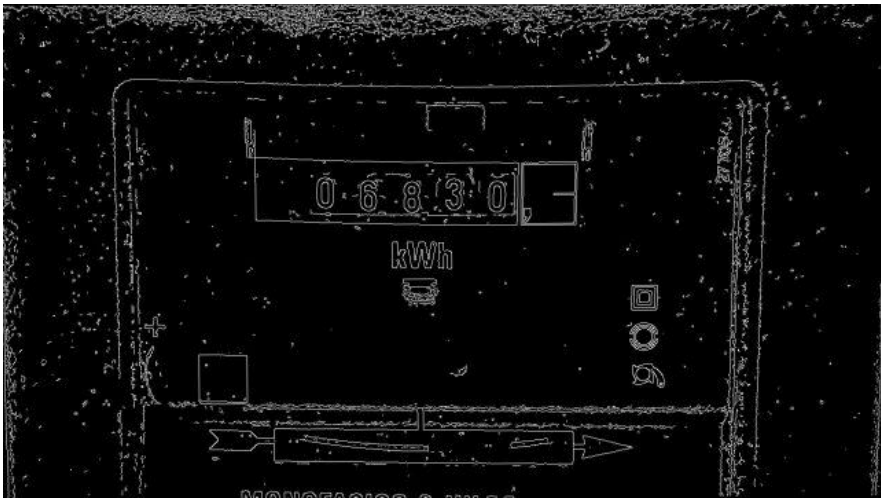


Figura 12: Imatge del comptador de llum en format Canny, orientada horitzontalment

- `contour_detection(canny_img)`: Aquesta funció rep com a paràmetre la imatge Canny aliniada horitzontalment. Llavors se li calculen els contorns de la imatge amb la funció `cv2.findCountours` [ref. 32][ref. 39]. Aquests contorns es guarden en la variable `contours`. Per a poder visualitzar els contorns torbats, aquests es dibuixen sobre la imatge original aliniada amb la funció `cv2.drawContours` [ref. 40] i es guarden a la imatge `contours.jpg` [Figura 13].

Un cop es tenen els contorns de la imatge, aquests es filtren per tal d' obtenir els contorns dels dígits. Per tal de fer això, per a cada contorn es pren la seva posició X i Y, la seva amplada i la seva alçada amb la funció `cv2.boundingRect` [ref. 41], i es comprova si aquests valors estan

dins del marge de les dimensions que hauria de fer el contorn d' un dígit. Si és el cas, es guarden els contorns filtrats a la llista `filtered_Contours`, i es guarden les posicions en la llista `digit_positions`. Recordem que els dígits estaran més o menys a la mateixa posició Y de la imatge, i per tant és important saber aquests valors. Els contorns filtrats es dibuixen sobre una imatge negre i es guarden amb el nom `filtered_contours.jpg` [Figura 14].

Figura 13: Imatge del comptador de llum amb els contorns trobats

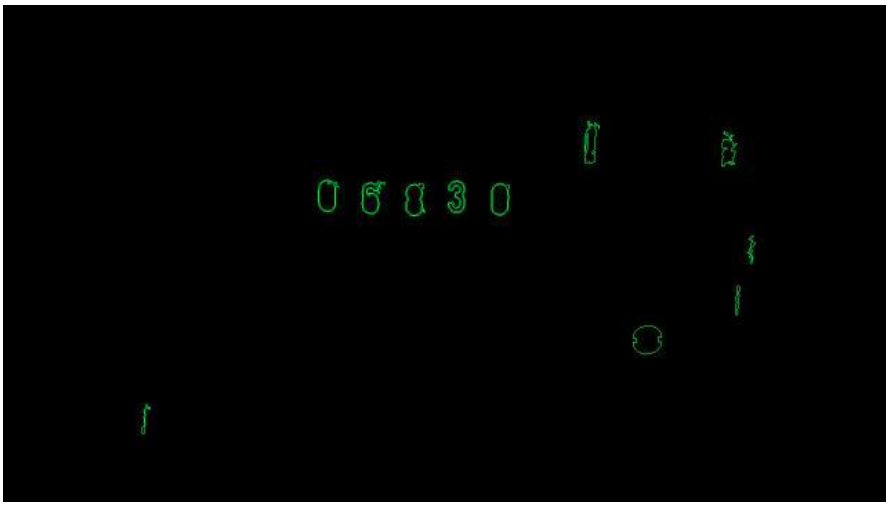


Figura 14: Contorns filtrats del comptador de llum

Un cop es tenen els contorns filtrats, es mira quina és la posició Y més repetida entre aquests contorns amb la funció `Counter.most_common` [ref. 42]. La posició Y resultant és la més probable de que sigui la posició Y on es troben els contorns dels dígits. Per tant, es fa un segon filtrat on es prenen els contorns que entren dins d' aquesta posició Y, amb un cert marge d' error. Els contorns es guarden a la llista `digitContours`. Es tornen a guardar les posicions X i Y a la

llista `digit_positions`, de manera ordenada. És possible que es detectin dos o més contorns a la mateixa posició, per tant es remouen els duplicats de la llista de posicions X i Y amb la funció `remove_duplicates`.

Un cop fets aquests passos de filtrat, tenim les posicions que haurien de correspondre als dígits. Per tant, per a cada posició X i Y, amb la funció `cv2.rectangle` [ref. 43] es dibuixa un rectangle de dimensions `w` i `h` corresponents a l' altura i amplada que hauria de tenir un dígit. Els rectangles trobats es guarden a la imatge `digits_found.jpg` [Figura 15]. Es retalla la imatge original per a cada rectangle i per a cada imatge retallada, en la qual hi hauria d' haver un dígit a cada una, se li aplica un thresholding, per tal de diferenciar el número del fons. Això es realitza amb la funció `cv2.adaptiveThreshold` [ref. 44], que consisteix en establir un threshold binari de valor de 0 a 255 i passar els píxels a blanc o a negre en funció de si sobrepassen o no el threshold. Com que els dígits són blancs i el fons negre, és fàcil realitzar aquest thresholding [Figura 16].

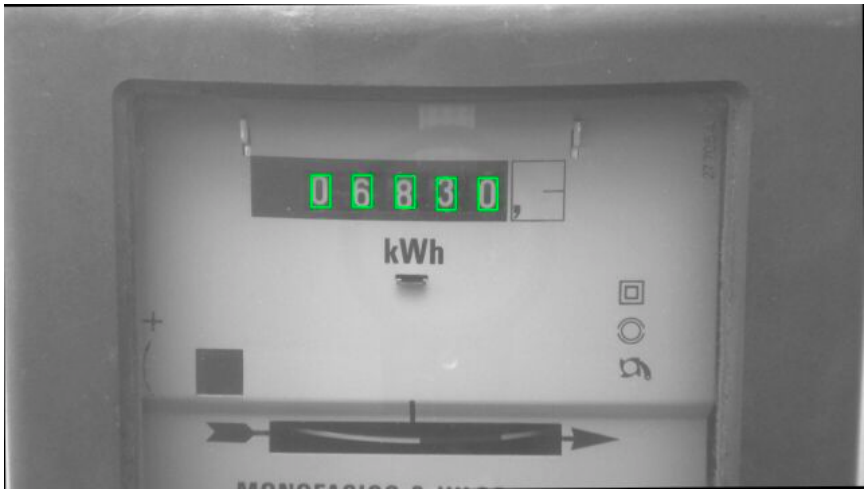


Figura 15: Posicions dels dígits trobats

Figura 16: Dígits del comptador, després d'aplicar-hi thresholding, preparats per a l'OCR

- `OCR_reading()`: Aquesta funció llegeix les imatges dels dígits que ha creat l' anterior funció `contour_detection`, una per una, i amb la funció `pytesseract.image_to_string` [ref. 45] amb la configuració `–psm 10 outputbase digits` [ref. 46], que interpreta la imatge com un sol dígit, fa el pas de imatge a string, i guarda cada string de dígit a un sol string, que és el que la funció retorna finalment.

Amb això s' acaba el processat d' imatge i s' obté l' string que a continuació s' enviarà via protocol MQTT [ref. 29] i es guardarà a la base de dades, juntament amb el timestamp i la imatge.

3.2.3.2. Processat d' imatge del comptador d' aigua

El processat d' imatge del comptador d' aigua és una mica més complicat. La primera raó és que, a part de fer el processat de dígits com amb el comptador d' aigua, també s' ha de processar la informació dels dials de les agulles, amb la qual cosa es requereix més processat i més passos a seguir. A més, tal i com es pot apreciar a la imatge del comptador [Figura 9], tant les agulles com un dels dígits són de color vermell, amb la qual cosa les seves formes no destaquen tant sobre el fons blanc i per tant en aplicar el Canny no es distingeixen adequadament.

Per a solucionar això, s' ha de fer una detecció del color vermell a la imatge, tal i com es mostrarà a continuació, per tal de fer-ne el processat de manera òptima.

El processat d' aquest comptador es realitza en els mòduls `needle_processing_water.py`, on es fa el processat de les agulles, i en el `OCR_processing_water.py`, on es duu a terme el processat dels dígits. Tots els passos es poden executar simultàniament executant el mòdul `full_water_processing.py`.

En el mòdul `OCR_processing_water.py` hi ha les mateixes funcions i es segueixen els mateixos passos de processat que els que s' han vist en el mòdul `OCR_processing.py`, a l' apartat **3.2.3.1.**, però s' implementa una nova funció, `red_ocr_value`, que s' encarrega de fer el processat del dígit vermell, ja que aquest segueix uns passos de processat diferents. Per fer-ho s' ajuda de la funció `red_detection` del mòdul `needle_processing_water.py` per a la detecció del color vermell, de la funció `contour_detection` per a trobar el contorn del dígit i de la funció `pytesseract.image_to_string` [ref. 45] per a passar el dígit torbat a un string.

Un cop executades les funcions del mòdul `OCR_processing_water.py`, s' obté l' string de dades corresponent als dígits del comptador, incloent el dígit vermell. Falta sumar-li les dades pertanyents a les agulles, que s' obtenen amb les funcions del mòdul `needle_processing_water.` A continuació s' expliquen els passos que es segueixen per al processat de les agulles:

- De la mateixa manera que amb el comptador de llum, es passa la imatge a format Canny i s' alinia horitzontalment de la mateixa manera que s' explica en l' apartat **2.2.3.1.**

- Un cop amb la imatge canny, ja rotada i retallada a la zona de les agulles, es fa la detecció del vermells. Això es realitza passant la imatge a format HSV (Hue, Saturation, Value) [ref. 47]. Aquest model de colors permet aïllar els colors aplicant una màscara a la imatge. D'

aquesta manera es poden aïllar les agulles de les imatges i trobar-ne bé la seva forma aplicant-hi un Canny edge detector.

- Un cop realitzat l' aïllament de les agulles, s' ha de buscar la seva posició en la imatge. Les agulles estan posicionades al centre d' uns dials que tenen forma circular. Per tal de trobar el centre de l' agulla, es realitza una detecció de cercles amb el mètode HoughCircles [ref. 48] a la imatge per tal de trobar aquests dials. Un cop trobats els cercles que corresponen als dials, es pot trobar el punt central d' aquests, el qual correspon aproximadament al centre de l' agulla.

- Un cop conegudes les posicions de les agulles, per a cada agulla s' ha de buscar el valor que marca en el dial. La manera en què això es fa és detectar les línies que tenen les agulles amb el mètode Houghlines [ref. 31], trobar-ne l' angle de desviació i la orientació (per tal de saber si l' agulla mira cap amunt o cap a baix), i convertir l' angle detectat a un número entre el 0.00 i el 9.99.

- Un cop es coneix, per a a cada agulla, el seu valor numèric, s' ajunten aquests valors en un string, que posteriorment s' ajuntaran a l' string de números extrets del marcador de dígits per formar el valor de lectura final que s' enviarà via protocol MQTT [ref. 29].

Com s' ha explicat, tots els passos descrits anteriorment es realitzen en les funcions del mòdul `needle_processing_water.py`, i tot el processat que s' ha explicat anteriorment així com també el del mòdul `ocr_processing_water.py` es pot realitzar executant directament el mòdul `full_water_processing.py`, que corre totes les funcions i retorna l' string de dades dels dígits.

Les funcions de `needle_processing_water.py` són les següents:

- `image_to_canny(img)`, `houghlines_detection(img, canny_img)`: Aquestes dues funcions realitzen la mateixa funció que la que s' explica en l' apartat **2.2.3.1**, per tant no cal entrar-hi en detall. La seva funció és passar la imatge a format Canny [Figura 17] i rotar la imatge per a aliniar-la horitzontalment i facilitar el processat. La imatge Canny aliniada es guarda sota el nom `water_edges_aligned.jpg` [Figura 19], i la imatge original rotada sota el nom `water_meter rotated.jpg` [Figura 18].

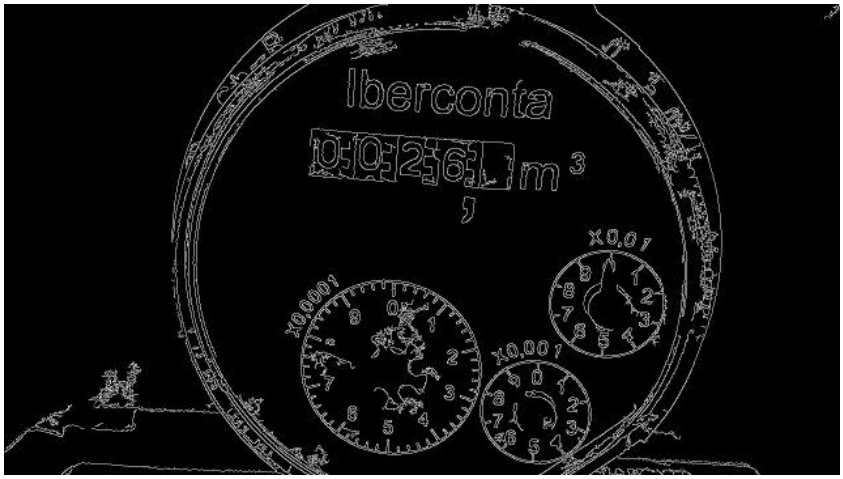
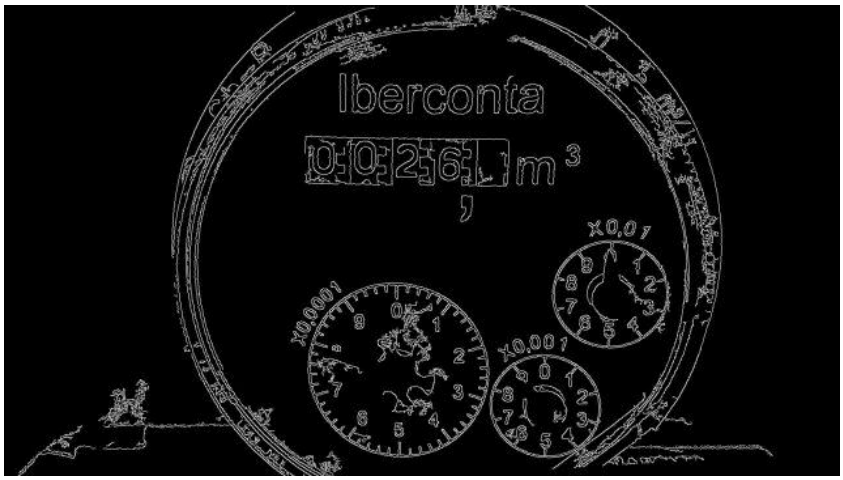


Figura 17: Imatge del comptador d' aigua en format Canny

Figura 18: Imatge de les línies horitzontals trobades en el comptador d' aigua amb el mètode HoughLines



-

Figura 19: Imatge del comptador d'0 aigua en format Canny, orientada horitzontalment

- `red_detection(cropped_img, needle=True)`: Els paràmetres d' aquesta funció són la imatge original retallada i un booleà amb valor per defecte `True`. Si `needle==True`, es crida la funció per a detectar les agulles de color vermell. Si `needle==False`, es crida la funció per a detectar el dígit de color vermell.

Per a la detecció de les agulles o del dígit, primerament es passa la imatge de format BGR a format HSV [ref. 47] amb la funció `cv2.cvtColor` [ref. 34] i el paràmatre `cv2.COLOR_BGR2HSV` [ref. 49]. A continuació es genera una màscara per a la imatge en format HSV, per tal d' aillar el color vermell. Per tal de fer això es defineixen dos arrays, `lower` i `upper`, que marquen els valors mínims i màxims de matís (Hue), saturació (Saturation) i lluminositat (Value) que deixa passar la màscara. Aquesta es genera amb la funció `cv2.inRange` [ref. 50]. Un cop s' obté la màscara, aquesta s' aplica a la imatge original retallada amb la funció `cv2.bitwise_and` [ref. 51], i s' obté la imatge amb les agulles o el dígit aïllat. La imatge es passa a format Canny per a poder detectar posteriorment les formes.

La imatge en format HSV [ref. 47] es guarda amb el nom `redhsv.jpg` [Figura 20], la imatge original amb la màscara aplicada es guarda amb el nom `red.jpg` [Figura 21], i les imatges Canny del dígit i de les agulles es guarden sota el nom de `decimal_edges.jpg` [Figura 22] i `needle_edges.jpg` [Figura 23] respectivament.

La funció retorna la imatge original en blanc i negre per a la posterior detecció dels dials de les agulles.

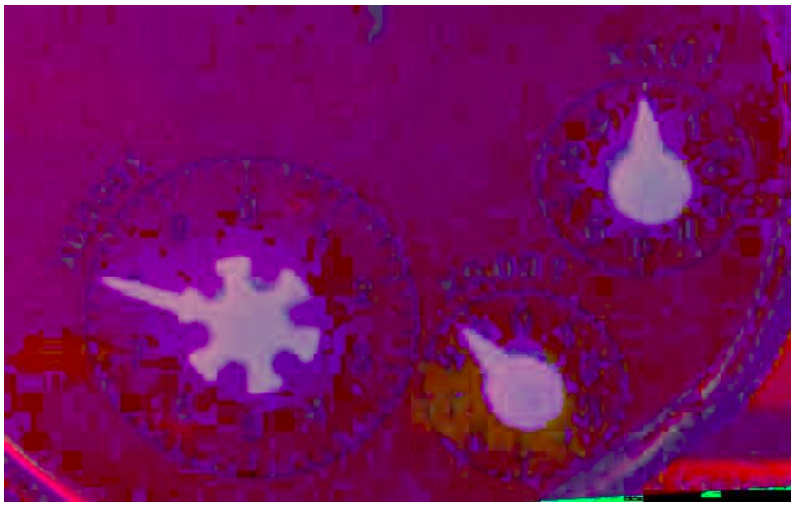


Figura 20: Imatge de les agulles del comptador d'aigua en format HSV

Figura 21: Imatge de les agulles del comptador després d'aplicar la màscara

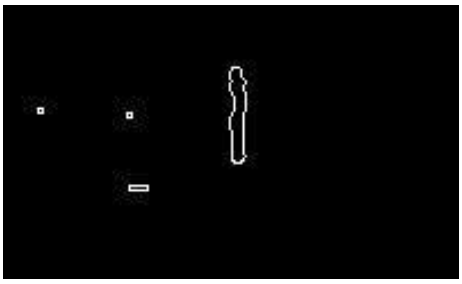


Figura 22: Contorn trobat del dígit vermell del comptador d'aigua.

Figura 23: Contorns trobat de les agulles del comptador d'aigua

- `dial_detection(img_copy_gray)`: Aquesta funció pren la imatge original en blanc i negre per tal de detectar els dials de les agulles. El primer pas que realitza és aplicar un thresholding a la imatge per tal de preparar-la per a la detecció dels cercles dels dials [Figura 24]. Un cop fet aquest pas, es realitza la detecció dels cercles amb l' ajuda de la funció `cv2.HoughCircles` [ref. 52] i el paràmetre `cv2.HOUGH_GRADIENT` [ref. 70]. La informació de cada cercle es guarda en un array de tuples on cada tupla té la posició X i Y del cercle i el radi R. Un cop es té aquest array, es pren la posició Y de cada cercle per tal d' identificar quin correspon a cada dial, i un cop conegut això, es retalla la imatge Canny de cada dial en funció de les dimensons del cercle. Aquestes imatges es guarden amb el nom `dial01.jpg`, `dial001.jpg` i `dial0001.jpg` [Figura 25]. Per a visualitzar els cercles que s' han trobat a la imatge, aquests es dibuixen sobre la imatge original amb la funció `cv2.circle` [ref. 69], i es guarda la imatge amb el nom `img_hough_circles.jpg` [Figura 26].

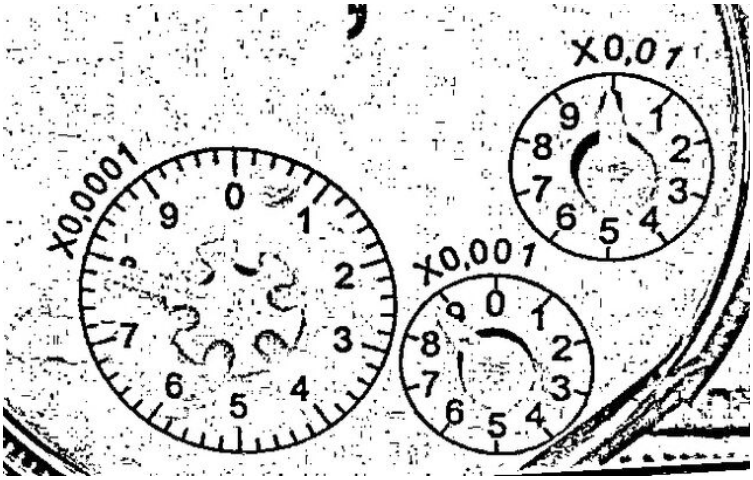


Figura 24: Imatge del comptador d'aigua a la zona de les agulles amb el thresholding aplicat

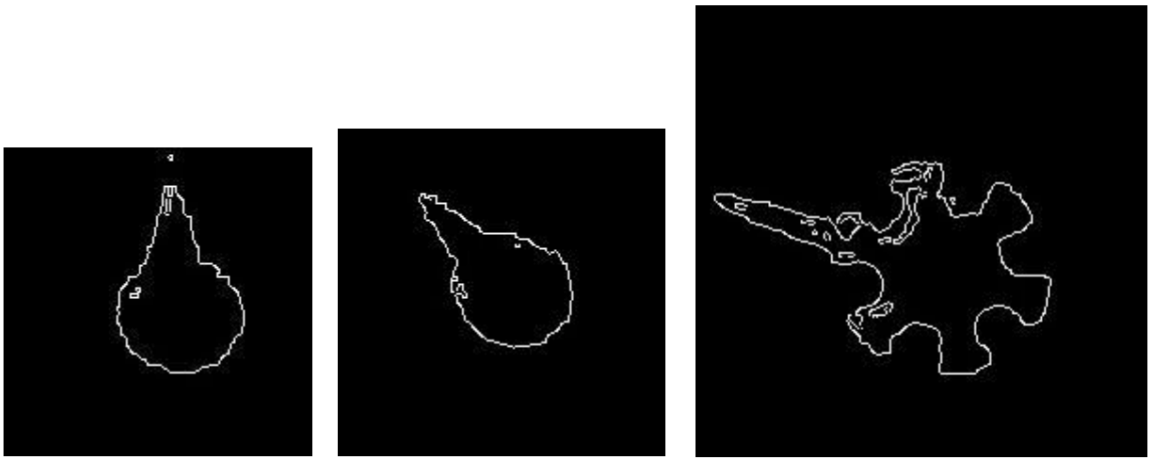


Figura 25: Imatge dels contorns de les agulles trobats

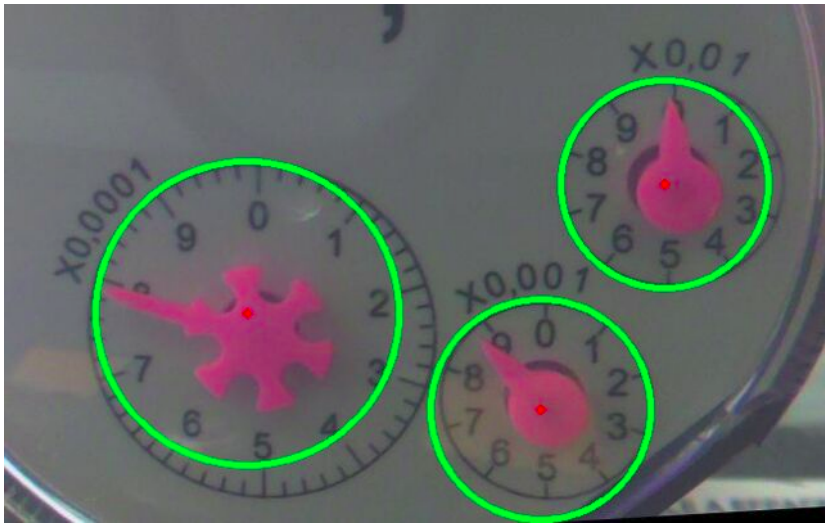


Figura 26: Imatge dels cercles corresponents als dials de les agulles trobats

- `needle_value()`: Aquesta funció és la més important, ja que s' encarrega de detectar les línies de les agulles, calcular-ne l' angle, i a partir de l' angle calcular-ne el valor que marca l' agulla en el dial. Per tal de fer això, per a cada imatge de les agulles que la anterior funció `dial_detection` ha creat, es segueixen els passos que es descriuen a continuació.

Primerament, es busquen les línies rectes que pertanyin a les agulles amb la funció `cv2.HoughLines` [ref. 36]. Aquesta funció crea un array de tuples amb informació sobre la rho i la theta, que representen la línia. Recordem que la línia es representa com $\rho = x\cos\theta + y\sin\theta$,on la rho és la distància perpendicular des de l' origen a la línia, i la theta és l' angle format per aquesta línia perpendicular i l' eix horitzontal, tal i com es mostra a la següent figura [Figura 27] [ref. 53]. Les línies detectades es dibuixen amb la funció `cv2.line` [ref. 54].

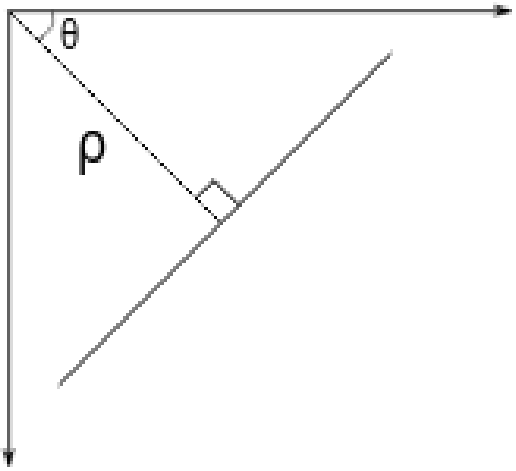


Figura 27: Representació gràfica del principi en què es basa el mètode HoughLines

Observem com coneixent rho i theta, queda una línia en funció de X i Y. La funció `needle_value` en calcula dos punts *x1*, *y1* i *x2*, *y2*. Coneixent aquests punts, es poden fer suposicions de quina direcció té la línia [Figura 28]. Per a fer-ho, s' han de tenir en compte dues coses:

- El valor de Y augmenta a mesura que baixa en la imatge (Augmenta de dalt a baix).
- El valor de X augmenta d' esquerra a dreta.

Coneixent això se sap que:

- Si *y2* > *y1*: En aquest cas, significa que la línia està en direcció descendent. Per tant la l' agulla ha d' estar en el segon quadrant del dial o en el quart. Per a saber-ho, es compara el valor de *y1* i *y2* amb el valor *y_center*, que és el valor de y del centre de la imatge. Si el valor de *y1* s' aproxima més al valor central, significa que l' agulla està en el quart quadrant, i si el valor de *y2* s' hi aproxima més, l' agulla està en el segon quadrant.
- Si *y2* < *y1*: La línia està en direcció ascendent. Per tant l' agulla ha d' estar en el primer quadrant o en el tercer. Es torna a comparar el valor de *y1* i *y2* amb el valor *y_center*. Si *y2* s' aproxima més a *y_center*, l' agulla està en el tercer quadrant. Si *y1* s' hi aproxima més, l' agulla està en el primer quadrant.

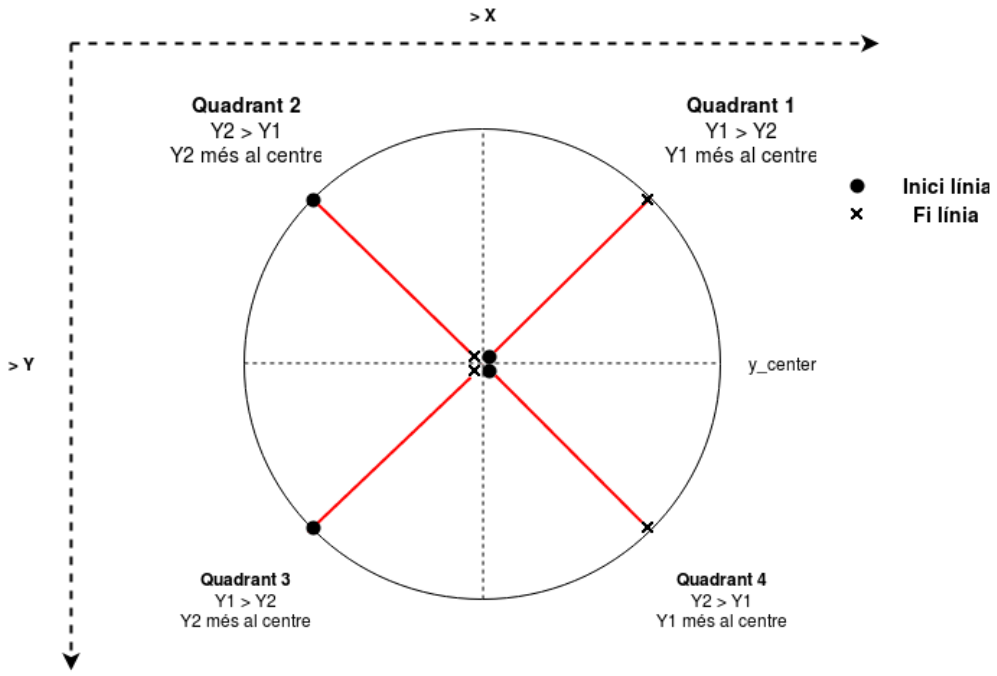


Figura 28: Esquema de criteri de detecció dels quadrants on es troben les agulles

Un cop es coneix en quin quadrant està l' agulla es pot calcular l' angle de l' agulla respecte l' eix horitzontal. L' angle es calcula amb la funció `np.arctan2` [ref. 55]. En funció del quadrant es calcula de forma diferent. A continuació es mostra com es realitza el càlcul de l' angle per a cada quadrant [Figura 29].

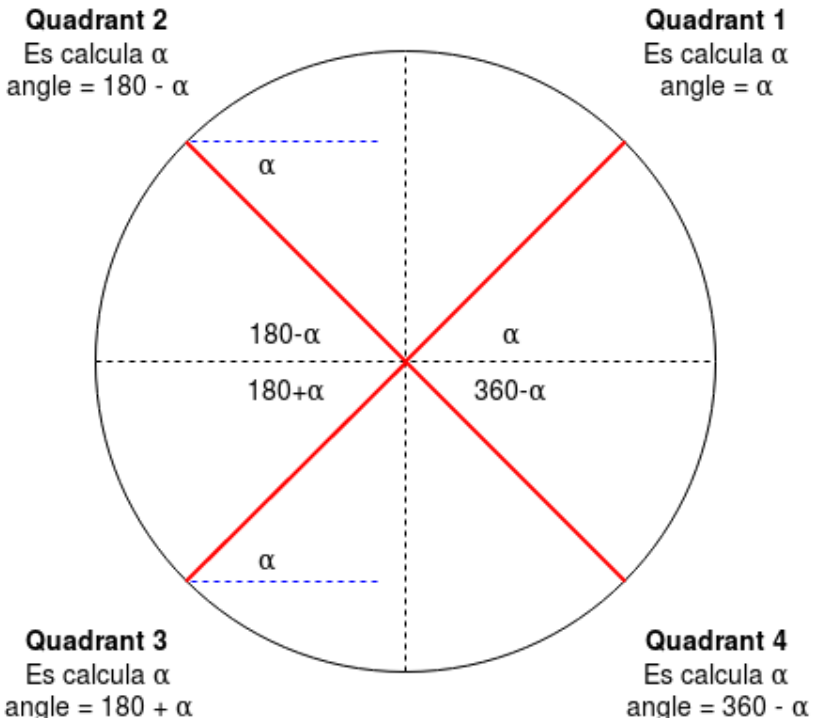


Figura 29: Esquema de criteri de detecció de l'angle les agulles

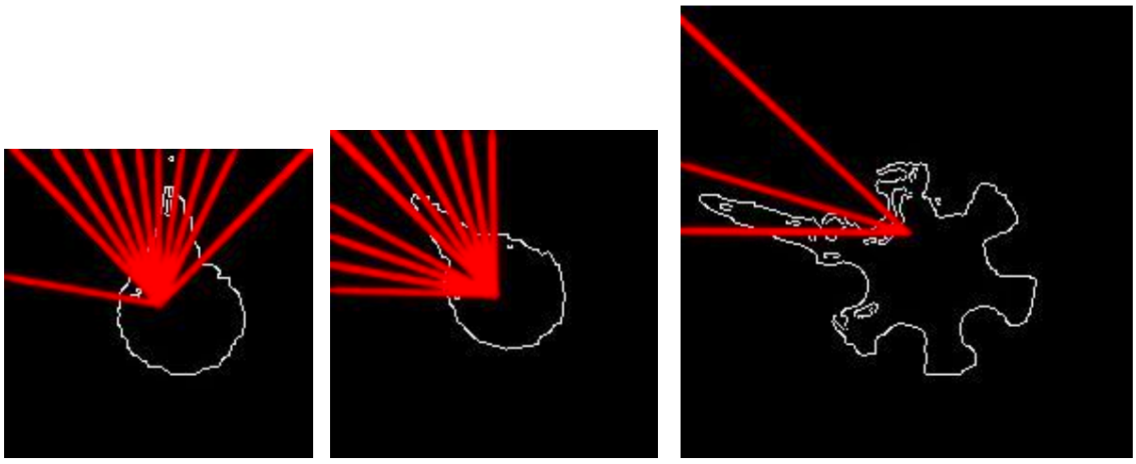


Figura 30: Línies de les agulles detectades amb el mètode HoughLines

En buscar les línies de les agulles, segurament apareixeran diverses línies amb una certa desviació [Figura 30]. Es calcula l' angle de totes les línies i es fa la mitjana de l' angle, i aquesta mitjana és el valor que es pren com a definitiu.

Un cop calculat l' angle de l' agulla respecte l' eix horitzontal, s' ha de fer la conversió de l' angle al número que marca en el dial. Es fa això prenent com a referència que el 0 es troba als 90 graus (eix vertical). A partir d' aquí, cada 3.6 graus més o menys, és una dècima més o menys. Es fa un sumatori fins que l' angle concorda i així s' aconsegueix el valor del dial.

Un cop conegut el valor de totes les agulles, aquests valors s' ajunten en un string, i posteriorment s' ajunten amb els valors que han donat els dígits amb el processat de les funcions del modul `ocr_processing_water.py`.

3.2.4. Enviament de les dades

Un cop obtingudes les dades fruit del processat d' imatge, aquestes s' han d' enviar al Host, per tal d' emmagatzemar-les. A continuació es mostrarà com s' ha implementat aquesta tasca.

3.2.4.1. El protocol MQTT i paho.mqtt

Per a la comunicació entre la màquina Host i la Raspberry Pi [ref. 8], s' ha establert el pas de missatges via protocol MQTT [ref. 29]. El protocol MQTT (Message Quering Telemetry Transport) és un protocol de pas de missatges que treballa per sobre del nivell de TCP/IP (A la capa d' aplicació com altres protocols com l' HTTP). Aquest protocol està pensat per a enviar missatges entre dispositius remots amb connexions on l' ample de banda de la xarxa és limitat, i també per a dispositius com sensors.

La estructura del protocol es basa en un patró de subscripció i publicació de tòpics, i dins d' aquests tòpics es produeix el pas de missatges. Per a rebre els missatges d' un determinat tòpic, el dispositiu client s' ha de subscriure al tòpic en qüestió, i rebrà els missatges quan algun altre dispositiu envïi un missatge amb el tòpic.

En el protocol MQTT [ref. 28] [Figura 31] existeixen tres entitats: el subscriptor, que es subscriu al tòpic, el publicador, que publica missatges dins del tòpic, i el broker, que és l' intermediari encarregat de distribuïr els missatges als clients interessats.

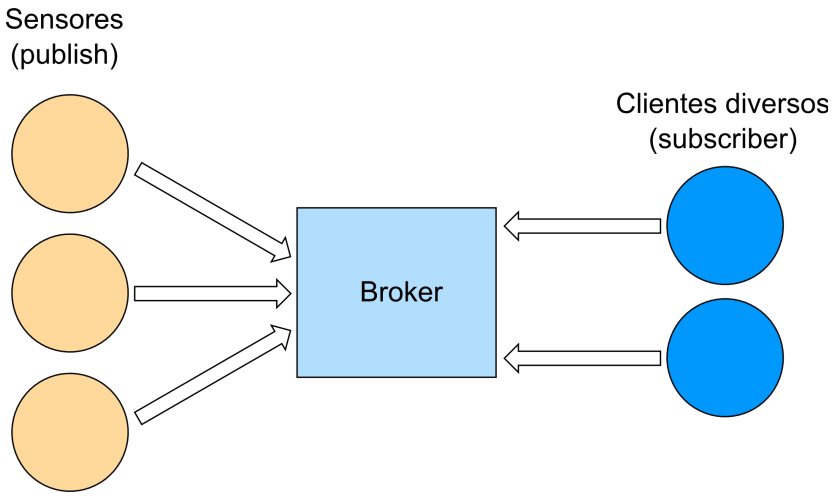


Figura 31: Principi de funcionament del protocol MQTT

Per a la implementació del protocol MQTT [ref. 28] en aquest projecte, s' ha utilitzat la llibreria `paho.mqtt` [ref. 17] de Python, la qual implementa les funcionalitats d' aquest protocol. Amb aquesta llibreria s' han pogut definir els tòpics i els missatges que s' han d' enviar en cada ocasió. Per tal de realitzar això, s' han establert dos clients: un és la màquina Host, que fa les requests tant de informació dels comptadors com per a modificar els paràmetres de la càmara. L' altre és la Raspberry Pi [ref. 8], que atèn les peticions de la màquina Host i envia les dades solicitades o bé modifica els paràmetres de la càmara. En el prototip es fa servir un broker de test disponible públicament que ofereix mosquitto [ref. 56], però es podria utilitzar un propi.

De la llibreria s' utilitzen les següents funcions:

- `mqtt.Client`: Per a crear el client del protocol [ref. 57].
- `connect`: Per a connectar-se al broker desitjat [ref. 58].
- `on_connect`: Per a establir què fer quan el client es connecta al broker. Per exemple, un cop connectat, el client es pot subscriure als tòpics que li interessen, i passar a esperar missatges [ref. 59].
- `on_publish`: Per a establir quines accions realitzar quan es publica un missatge [ref. 60].
- `on_message`: Per a establir quines accions realitzar quan es rep un missatge. Aquí s' atendran els diferents tipus de missatge i s' actuarà en conseqüència [ref. 61].

- `on_subscribe`: Per a establir quines accions realitzar quan el client s' ha subscrit a un tòpic. Per exemple, en el Host (`sub.py`), es mira quin tipus de request se li està demanant i passa a publicar-la, per tal de rebre els missatges de tornada dels tòpics que s' ha acabat de subscriure [ref. 62].

3.2.4.2. Estructura de pas de missatges

Per a la implementació del pas de missatges s' han creat dos mòduls: `sub.py`, que actua com a client per la part de la màquina HOST, fa les peticions de dades o de modificació de paràmetres de la càmara a la Raspberry Pi [ref. 8], rep les dades dels comptadors i s' encarrega de guardar-les a la base de dades.

L' altre, `pub.py`, actua com a client per part de la Raspberry Pi , rep les peticions del client de la màquina Host, interactua amb la càmara per a que capturi les imatges, executa les funcions de processat d' imatge, envia la imatge, el nom (timestamp) i la dada de lectura, i també modifica els paràmetres de la càmara si rep la petició corresponent.

L' estructura que s' ha implementat per al pas de missatges és la següent:

HOST

A `sub.py` (Host), el client Host es subscriu als següents tòpics:

- `ocr_image_name`: tòpic per tal de rebre el nom de la imatge en forma de timestamp (string).
- `ocr_image`: tòpic per tal de rebre la imatge en forma de BLOB [ref. 28].
- `ocr_data`: tòpic per a rebre les dades de lectura del comptador (string).

El Host envia missatges amb els següents tòpics:

- `ocr_request`: quan el client vol informació del comptador, envia un missatge amb aquest tòpic. El cos del missatge és:
    - "1": Si es vol rebre informació del comptador d' aigua.
    - "2": Si es vol rebre informació del comptador de llum.

Quan s' envia un `ocr_request`, s' esperen com a resposta tres missatges amb els tòpics `ocr_image_name`, `ocr_image`, `ocr_data`, contenint la informació descrita anteriorment. Quan es reben, es guarda el seu contingut a la base de dades.

- `cam_request`: quan el client vol modificar els paràmetres de la càmara, envia un missatge amb aquest tòpic. El cos del missatge és un string amb els valors de cada paràmetre, separats per una coma (“`,`”).

  Quan s' envia un `cam_request` no s' espera resposta per part de la Raspberry Pi [ref. 8].

RASPBERRY

A `pub.py`, el client de la Raspberry Pi es subscriu als següents tòpics:

- `ocr_request`: tòpic per tal de rebre la petició de informació del comptador d' aigua (cos de missatge=”`1`”) o de llum (cos de missatge=”`2`”). Quan es rep, s' executen les funcions de processat corresponents, d' on s' extreuran les dades de lectura del comptador.

- `cam_request`: tòpic per tal de modificar els paràmetres de la càmara. Quan es rep, s' extreuen els paràmetres sol·licitats del missatges i es crida la funcio `resetup` del mòdul `camera_operations.py` amb els nous paràmetres.

La Raspberry Pi envia missatges amb els següents tòpics:

- `ocr_image_name`: tòpic per tal d' enviar el nom de la imatge en forma de timestamp (string).

- `ocr_image`: tòpic per tal d' enviar la imatge en forma de BLOB [ref. 28].

- `ocr_data`: tòpic per a enviar les dades de lectura del comptador (string).

  El timestamp i la imatge són valors que retorna la funció `shoot` del mòdul `camara_operations`, que es crida just quan es rep una petició de tipus `ocr_request`. La `ocr_data` (dades de lectura del comptador) és el què retorna l' execució de les funcions de processat d' imatge.

La estructura en forma gràfica es mostra a continuació [Figura 32].

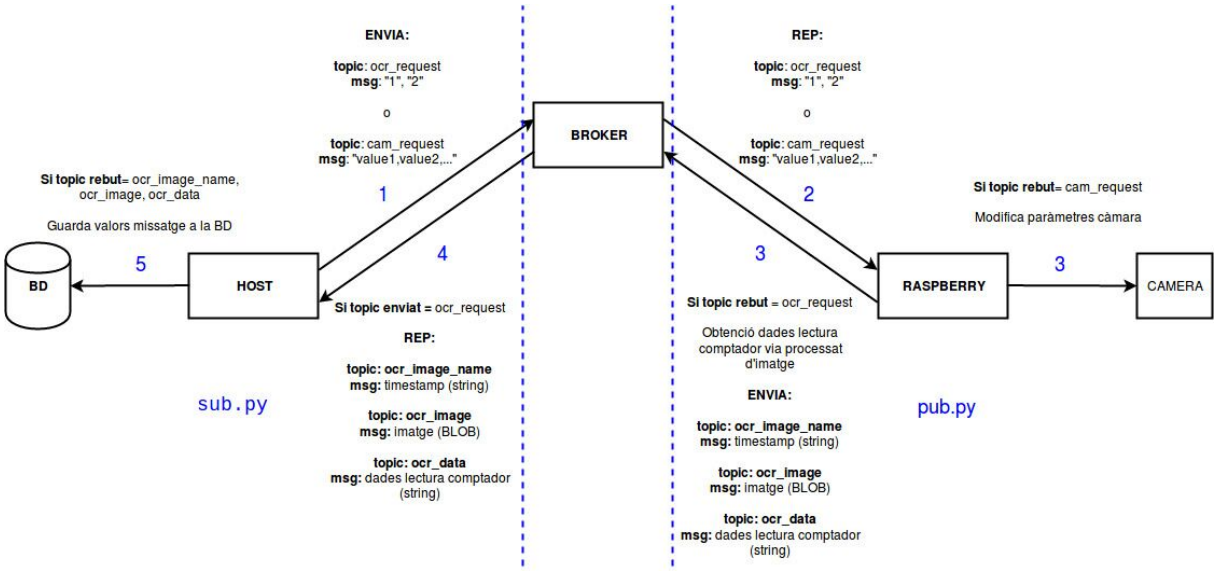


Figura 32: Esquema de l'estructura del pas de missatges entre el Host i la Raspberry via MQTT

3.2.4.3. Execució i temporització de les peticions

Les peticions es realitzen executant l' script `sub.py` per la part del Host. La Raspberry Pi [ref. 8] ha de tenir el script `pub.py` executant-se contínuament i esperant rebre missatges.

Per tal de diferenciar quin tipus de request ha de realitzar el Host, a l' hora d' executar l' script de Python per línia de comandes s' ha de passar un d' aquests valors:

- 1: per a una petició de dades del comptador d' aigua.
- 2: per a una petició de dades del comptador de llum.
- 3: per a una petició de modificació dels paràmetres de la càmara.

L' script s' executa a través de tres scripts de shell, els quals criden la funció amb l' argument de línia de comandes. Els fitxers són `exec1.sh`, `exec2.sh` i `exec3.sh`.

Per exemple, el contingut de `exec1.sh` és:

```
#!/bin/sh

set -x

python /home/<user>/<PATH_TO_SCRIPT>/sub.py 1
```

Per tal de realitzar l' execució periòdica d' aquests scripts, es fa ús de Cron [ref. 19]. Es defineix una taula on s' especifica la periodicitat amb la qual s' ha d' executar el script de shell. Executant `crontab -e` es pot editar la taula. Per exemple, per a executar l' script cada 5 minuts:

```
# m h  dom mon dow   command

/5  * * * /home/<user>/<PATH_TO_SCRIPT>/exec1.sh
```

3.2.5 Emmagatzemament de les dades

Les dades que es reben a través dels missatges es guarden a la base de dades de nom `app.bd`. Com s' ha esmentat anteriorment, les dades que s' han de guardar són la imatge en format BLOB [ref. 28], el timestamp, que serveix com a identificador únic de la imatge, i la dada extreta del comptador.

La base de dades té dues taules, `images_llum` i `images_aigua`, una per a emmagatzemar les dades de cada comptador. L' estructura de la base de dades es defineix en el fitxer `images.sql`, el qual té el següent contingut:

```
CREATE TABLE IF NOT EXISTS images_LLUM( nom timestamp PRIMARY KEY NOT
  NULL,

imatge blob,

OCR text

   );


CREATE TABLE IF NOT EXISTS images_AIGUA( nom timestamp PRIMARY KEY NOT
  NULL,

imatge blob,

OCR text

   );
```

4. TRACTAMENT DE LES DADES

Un cop es tenen les dades emmagatzemades a la base de dades, es pot passar al tractament d' aquestes. Per tal de fer això, és convenient poder passar les dades a un fitxer `csv`, per tal que l' usuari pugui veure les dades fàcilment i poder-ne fer gràfics, càlculs de consum energètic, etc …

Per tal de fer això, s' ha implementat el mòdul `db_operations.py`, que l' usuari pot utilitzar per passar la informació de la base de dades a un fitxer `csv`, i també per a guardar les imatges de la base de dades a un fitxer `jpg`.

4.1. Visualització de la imatge

Recordem que la base de dades guarda tres dades de cada comptador: el timestamp, que identicica la imatge i mostra la data en què s' ha près la imatge, la imatge en format BLOB [ref. 28], i la lectura del comptador.

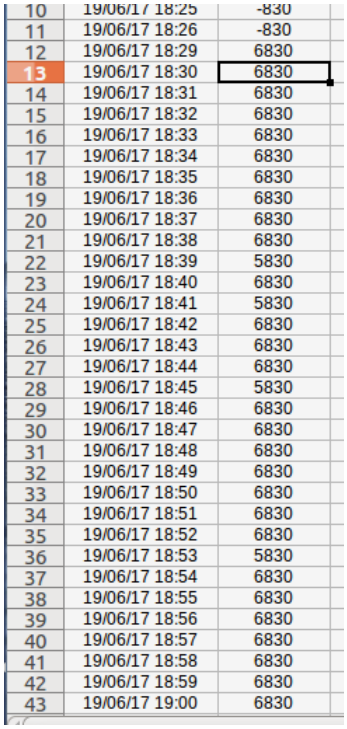
Per tal de poder passar la imatge de format BLOB [ref. 28] a un fitxer `jpg`, el mòdul `db_operations.py` implementa la funció `writeImatge(meter_type)`, on `meter_type` indica el tipus de comptador que l' usuari vol que es guardin les imatges, i que pot prendre els valors `'water'` o `'electricity'`.

Les imatges es guardaran amb el seu timestamp com a nom.

4.2. Extracció i tractament de les dades

L' altra funcionalitat que implementa el mòdul `db_operations.py` és la de guardar les dades de la base de dades en un fitxer `csv`. D' això se n' encarrega la funció `to_CSV(meter_type)`, on `meter_type` també indica el tipus de comptador que l' usuari vol que es guardin les dades, i que pot prendre els valors `'water'` o `'electricity'`. Els fitxers csv que es crearan seran el `water_data.csv` i el `electricity_data.csv`. Per tal de guardar les dades en el fitxer `csv`, la funció `to_CSV` fa ús de la llibreria `csv` [ref. 63] de Python. La informació es guarda en format Unicode UTF-8 [ref. 64], així que per tal de visualitzar correctament les dades, s' ha d' obrir el fitxer de manera que s' interpretin els caràcters en aquest format.

A continuació es pot veure una demostració del contingut d' aquests fitxers [Figura 33].



10	19/06/17 18:25	-830
11	19/06/17 18:26	-830
12	19/06/17 18:29	6830
13	19/06/17 18:30	6830
14	19/06/17 18:31	6830
15	19/06/17 18:32	6830
16	19/06/17 18:33	6830
17	19/06/17 18:34	6830
18	19/06/17 18:35	6830
19	19/06/17 18:36	6830
20	19/06/17 18:37	6830
21	19/06/17 18:38	6830
22	19/06/17 18:39	5830
23	19/06/17 18:40	6830
24	19/06/17 18:41	5830
25	19/06/17 18:42	6830
26	19/06/17 18:43	6830
27	19/06/17 18:44	6830
28	19/06/17 18:45	5830
29	19/06/17 18:46	6830
30	19/06/17 18:47	6830
31	19/06/17 18:48	6830
32	19/06/17 18:49	6830
33	19/06/17 18:50	6830
34	19/06/17 18:51	6830
35	19/06/17 18:52	6830
36	19/06/17 18:53	5830
37	19/06/17 18:54	6830
38	19/06/17 18:55	6830
39	19/06/17 18:56	6830
40	19/06/17 18:57	6830
41	19/06/17 18:58	6830
42	19/06/17 18:59	6830
43	19/06/17 19:00	6830

Figura 33: Dades de la base de dades passades a format csv

Per tal de guardar les dades en aquests fitxers de forma periòdica i autònoma, es pot configurar Cron per tal que executi els script de shell `water_to_csv.sh` o `electricity_to_csv.sh`, que cridaran la funció `to_csv`.

5. LLISTAT DE COSTOS DEL PROTOTIPUS

En aquest apartat es realitza el càlcul del cost total del prototipus. Cal remarcar que només es té en compte el cost dels components físics del sistema, i per tant no s' ha de confondre amb el possible seu possible de venda, ja que aquí no s' estudien els costos de programació i altres costos associats.

En la taula següent es mostren tots els components físics del sistema i es seu cost.

Component	Cost (euros)
Kit de Raspberry Pi 3 [ref. 65]	76.90
Pi Camera Noir V2.1 [ref. 66]	29.96
4 Leds [ref. 67]	2
Impressió 3D de la montura	16
Conjunt de cargols i femelles [ref. 68]	11

Per tant, el cost total del sistema físic és de 135.86 euros.

6. CONDICIONS PER A L' EXECUCIÓ DEL SISTEMA

Per tal d' utilitzar el sistema de manera correcta, l' usuari ha de conèixer alguns aspectes sobre aquest, que es detallaran a continuació.

6.1. Posada en marxa i correcte funcionament del sistema.

Per a posar en marxa el sistema correctament, s' han de complir els següents requisits:

- Tenir la montura col·locada correctament en el comptador, de manera que la càmara tingui en el seu camp de visió les dades que s' han de prendre del comptador, ja sigui de manera directa o través del mirall que hi ha al suport.

- Que la montura del comptador es trobi en una zona amb connexió Wi-Fi, per tal que la Raspberry Pi [ref. 8] pugui rebre les peticions de presa de dades, així com transmetre les dades.

- Tenir la Raspberry Pi alimentada correctament.

- Si el comptador es troba en una zona poc iluminada o és de nit, fer ús dels LEDs per a iluminar-lo.

- Per la part del Host, tenir-lo configurat de manera que pugui establir connexió amb la Raspberry Pi [ref. 8], i tenir la taula de Cron configurada correctament per a fer la petició de dades amb el comptador que es desitgi, i també per guardar les dades de la base de dades als fitxers `csv`, si així es desitja.

- Utilitzar una lent adequada per tal que la càmara pugui prendre imatges nítides. D' altra manera el processat d' imatge serà impossible. A l' apartat **6.2.2.** es tracta aquest tema.

6.2. Implementacions extres del sistema.

6.2.1. Els miralls

Els miralls [Figura 34] s' utilitzen per tal de poder realitzar captures del comptador de manera indirecta i des d' un angle més tancat, i així optimitzar al màxim l' espai que ocupa la montura. Aquest és un dels avantatges que suposa fer ús dels miralls.

Tot i així s' han de tenir en compte alguns aspectes a l' hora de fer ús dels miralls. El primer és que, si es captura la imatge dels dígits a través dels miralls, aquesta imatge estarà cap per avall, i per tant s' haurà de rotar 180 graus, i voltejar-la horitzontalment. Per tal de facilitar les coses per a l' usuari, aquest pot fer ús del canvi de paràmetres de la càmara [ref. 5] que ofereix el protocol de pas de missatges entre el Host i la Rapberry, tal i com s' ha vist en l' apartat **3.2.4.2**.

Figura 34: Mirall colocat al suport

Un aspecte important a tenir en compte és que, en el comptador d' aigua, si es prenen imatges a través del mirall, no és possible el processat de les agulles, només dels dígits. Per tant, s' ha d' adaptar el sistema per al processat del comptador d' aigua sense agulles. Això comporta una pèrdua de dades. Tot i així, les agulles marquen els dígits menys significatius, així que la pèrdua d' aquests dígits no comporta una pèrdua d' informació molt significativa.

6.2.2. La lent

Durant la realització d' aquest projecte s' ha vist que la càmara [ref. 5] no és capaç d' enfocar gaire bé a distàncies curtes. Aquest sistema requereix capturar imatges a una distància força petita (uns 10 centímetres). La càmara no és capaç d' autoenfocar-se, i per tant s' ha requerit fer ús d' una lent externa de 12 dioptries. Per tal de conèixer la força necessària de la lent, s' ha fet ús de la fòrmula [ref. 76] que es mostra a continuació:

$$\frac{1}{f} = \frac{1}{s} + \frac{1}{s'}$$

On *f* és la inversa de la força de la lent, *s* és la distància entre la lent i el comptador, i *s'* és la distància focal, que és la distància on la càmara és capaç d' enfocar.

Sabent que la distància *s* és de 10 centímetres, i que la càmara enfoca a uns 50-75 centímetres (*s'*), es pot trobar la força 1/*f* necessària de la lent, que en aquest cas és d' entre 11,33 i 12 dioptries.

7. TESTEIG DEL SISTEMA

A continuació es mostrarà com s' han realitzat un conjunt de testejos del sistema, els resultats que s' han extret d' aquests i les conclusions que se' n poden extreure.

7.1. Testejos amb el comptador de llum

7.1.1. Càmara enfocant directament al comptador

En aquest test, fem que el sistema faci lectures del comptador de llum durant 30 minuts. El comptador no està connectat a la corrent i per tant no donarà valors diferents durant aquest temps, però es comprovarà la robustesa del sistema, ja que durant les proves s' ha vist que el processat d' imatge és força delicat. El mínim canvi, ja sigui en la orientació de la càmara, la lluminositat, etc... afecta al processat. Per tant, la intenció d' aquest test és comprovar que el sistema pot actuar de forma independent durant un cert temps donant resultats correctes, malgrat la fragilitat del processat d' imatge.

El valor que marquen els dígits del comptador és 06830 [Figura 35].

Figura 35: Montatge del sistema amb el comptador de llum

Per tal de fer això, comprovem primer que el sistema és capaç de rebre una petició des del Host, capturar la imatge, processar-la i enviar les dades i la imatge a la base de dades del Host.

Per tal de fer això cridem `sub.py` manualment des del Host. A la Raspberry Pi [ref. 8] s' escolten les peticions ja que `pub.py` s' està executant [Figura 36] [Figura 37].

```
user@TOSHIBA:~/Escritorio/QUART/TFG/rasp$ python sub.py 1
Connect result flag: 0
Connection succeded
Unknown command.
Request sent.
Topic: ocr_image_name
Image name: 2017-06-19 17:31:15
Topic: ocr_image
Image received
Image length: 215380
<type 'str'>
Image file saved
Topic: ocr_data
OCR data received: 06830
Light meter image and OCR reading saved in database
```

Figura 36: Flux d' execució de sub.py

```
pi@raspberrypi:~/Desktop/TFG/Scripts $ python3 pub.py
Connect result flag: 0
Connection succeded
Topic:  ocr_request
Request:  1: OCR light meter image processing.
Detected lines angle:

-3.01128087031
-3.01128087031
Horizontal angle deviation: -3.01128087031

Number of contours found: 60

Number of filtered contours found: 17

Digits X positions not filtered: [467, 468, 375, 376, 389, 390,
 306, 307, 555, 556, 564, 565, 222, 223, 230, 471, 472]

Digits Y positions not filtered: [200, 201, 199, 200, 93, 94, 9
2, 93, 91, 93, 99, 100, 91, 92, 99, 90, 91]

Most probable digit y position: 91

Number of digit contours found: 13 (should be 6)

Digits positions: [[222, 91], [223, 92], [230, 99], [306, 92],
[307, 93], [389, 93], [390, 94], [471, 90], [472, 91], [555, 91
], [556, 93], [564, 99], [565, 100]]

Digits positions (X not repeated, Y filtered): [[222, 91], [306
, 92], [389, 93], [471, 90], [555, 91]]

Digits positions (filtered): [[222, 91], [306, 92], [389, 93],
[471, 90], [555, 91]]
Number of filtered contours found: 17

Digits X positions not filtered: [467, 468, 375, 376, 389, 390,
 306, 307, 555, 556, 564, 565, 222, 223, 230, 471, 472]

Digits Y positions not filtered: [200, 201, 199, 200, 93, 94, 9
2, 93, 91, 93, 99, 100, 91, 92, 99, 90, 91]

Most probable digit y position: 91

Number of digit contours found: 13 (should be 6)

Digits positions: [[222, 91], [223, 92], [230, 99], [306, 92],
[307, 93], [389, 93], [390, 94], [471, 90], [472, 91], [555, 91
], [556, 93], [564, 99], [565, 100]]

Digits positions (X not repeated, Y filtered): [[222, 91], [306
, 92], [389, 93], [471, 90], [555, 91]]

Digits positions (filtered): [[222, 91], [306, 92], [389, 93],
[471, 90], [555, 91]]

Digits interpreted from the image:

0
6
8
3
0
Light meter reading: 06830
Image length: 215380
Image sent.
```

Figura 37: Flux d' execució de pub.py durant el processat del comptador de llum.

Com podem observar, s' ha fet la petició d' obtenir les dades del comptador de llum, la Raspberry Pi ha fet el processat, ha enviat les dades i la imatge, i el Host ha guardat les dades a la base de dades.

Per tal de comprovar que això pot funcionar independentment, modifiquem la taula Crontab [ref. 19][Figura 38], per tal que la acció que s' ha vist anteriorment es produeixi periòdicament, en aquest cas cada minut durant 30 minuts.

```
# m h  dom mon dow   command
#*/1 * * * * /home/user/Escritorio/QUART/TFG/rasp/exec1.sh;
```

Figura 38: Configuració de la taula Cron per al testeig

Ara el Host farà una petició de dades cada minut.

Després de 30 minuts, es comprova si les dades s' han guardat a la base de dades, a la taula `images_LLUM`. Tal i com es veu [Figura 39], així és. El primer camp és el nom de la imatge (data de quan s' ha près la imatge), el segon camp conté la imatge en forma de stream de bytes [ref. 28], i el tercer camp la lectura OCR.

```
2017-06-19 18:29:04|[illegible]06830
2017-06-19 18:30:04|[illegible]06830
2017-06-19 18:31:04|[illegible]06830
2017-06-19 18:32:05|[illegible]06830
2017-06-19 18:33:04|[illegible]06830
2017-06-19 18:34:03|[illegible]06830
2017-06-19 18:35:03|[illegible]06830
2017-06-19 18:36:04|[illegible]06830
2017-06-19 18:37:03|[illegible]06830
2017-06-19 18:38:03|[illegible]06830
2017-06-19 18:39:03|[illegible]05830
2017-06-19 18:40:03|[illegible]06830
2017-06-19 18:41:04|[illegible]05830
2017-06-19 18:42:04|[illegible]06830
2017-06-19 18:43:03|[illegible]06830
2017-06-19 18:44:04|[illegible]06830
2017-06-19 18:45:04|[illegible]05830
2017-06-19 18:46:03|[illegible]06830
2017-06-19 18:47:03|[illegible]06830
2017-06-19 18:48:03|[illegible]06830
2017-06-19 18:49:03|[illegible]06830
2017-06-19 18:50:04|[illegible]06830
2017-06-19 18:51:03|[illegible]06830
2017-06-19 18:52:04|[illegible]06830
2017-06-19 18:53:04|[illegible]05830
2017-06-19 18:54:04|[illegible]06830
2017-06-19 18:55:04|[illegible]06830
2017-06-19 18:56:04|[illegible]06830
2017-06-19 18:57:04|[illegible]06830
2017-06-19 18:58:04|[illegible]06830
2017-06-19 18:59:03|[illegible]06830
2017-06-19 19:00:03|[illegible]06830
sqlite> _
```

Figura 39: Contingut de la base de dades després del primer testeig

Com s' oberva, les dades obtingudes són generalment correctes, però en algun cas hi ha un dígit que no ha donat el valor que havia de donar.

Es realitza un segon testeig, després de retocar alguns elements del codi, i observem quins són els resultats [Figura 40].

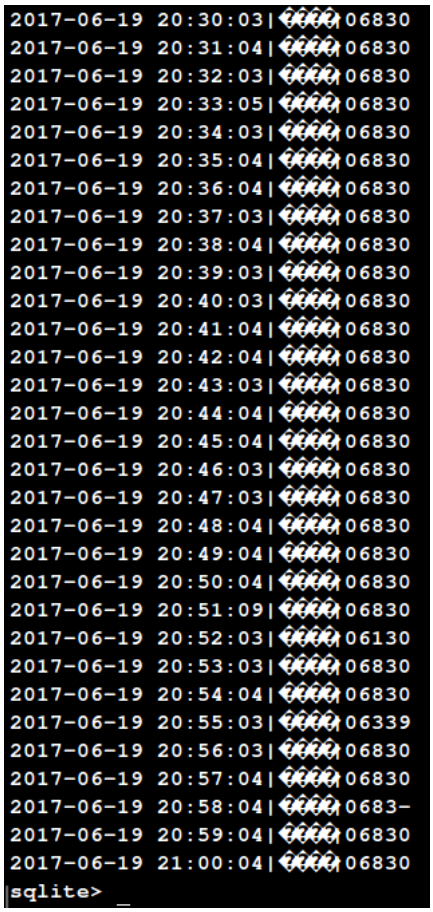


Figura 40: Contingut de la base de dades després del segon testeig

Es pot comprovar que hi ha hagut també algun error. Tot i així, la majoria de mostres segueixen sent bones. Es veu que de mitja hi ha uns 3-4 errors per cada 30 mostres. Per tant, el sistema dóna uns valors correctes el 90% de les vegades, aproximadament.

Un cop es tenen els valors a la base de dades, aquests es poden visualitzar en un fitxer `csv`, tal i com s' explica en l' apartat **4.2.** Fent ús de l' script `electricity_to_csv.sh`, es passen les dades al fitxer `electricity_data.csv` [Figura 41].

97	19/06/17 20:29	-685
98	19/06/17 20:30	6830
99	19/06/17 20:31	6830
100	19/06/17 20:32	6830
101	19/06/17 20:33	6830
102	19/06/17 20:34	6830
103	19/06/17 20:35	6830
104	19/06/17 20:36	6830
105	19/06/17 20:37	6830
106	19/06/17 20:38	6830
107	19/06/17 20:39	6830
108	19/06/17 20:40	6830
109	19/06/17 20:41	6830
110	19/06/17 20:42	6830
111	19/06/17 20:43	6830
112	19/06/17 20:44	6830
113	19/06/17 20:45	6830
114	19/06/17 20:46	6830
115	19/06/17 20:47	6830
116	19/06/17 20:48	6830
117	19/06/17 20:49	6830
118	19/06/17 20:50	6830
119	19/06/17 20:51	6830
120	19/06/17 20:52	6130
121	19/06/17 20:53	6830
122	19/06/17 20:54	6830
123	19/06/17 20:55	6339
124	19/06/17 20:56	6830
125	19/06/17 20:57	6830
126	19/06/17 20:58	-683
127	19/06/17 20:59	6830
128	19/06/17 21:00	6830

Figura 41: Extracte del fitxer `electricity_data.csv`

7.1.2. Càmara enfocant al mirall

En aquest cas, realitzarem el testeig de la presa de dades del comptador de llum amb la càmara enfocant al mirall, el qual reflecteix la lectura del comptador [Figura 42].

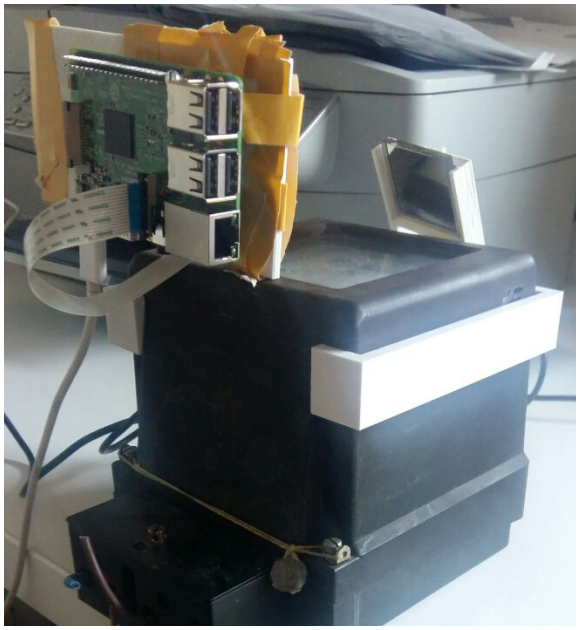


Figura 42: Montatge del sistema amb el comptador de llum, amb la càmara enfocant el mirall

El testeig també consistirà en realitzar una petició de dades cada minut, durant 30 minuts, per tal de comprovar si el processat d' imatge és consistent.

Després d' haver adaptat el codi per tal que el processat funcioni adequadament amb la imatge reflectida del mirall, es procedeix a executar `pub.py`, el qual escolta les peticions del Host, que crida `sub.py` periòdicament. Un aspecte important és que, a l' hora de fer el setup de la càmara, se li ha d' establir un paràmatre de rotació de 180 graus [ref. 24] i voltejar la imatge horitzontalment [ref. 74], ja que en la imatge, els dígits estaran al revés.

Observem com el sistema és capaç de prendre la imatge [Figura 43], detectar-ne la desviació, rotar-la [Figura 44], detectar els contorns dels dígits [Figura 45] [Figura 46], i passar-los a text.

Figura 43: Captura presa de la reflexió del mirall per part de la càmara, ja voltejada i orientada correctament

Figura 44: Imatge orientada hortizontalment

Figura 45: Contorns filtrats corresponents als dígits

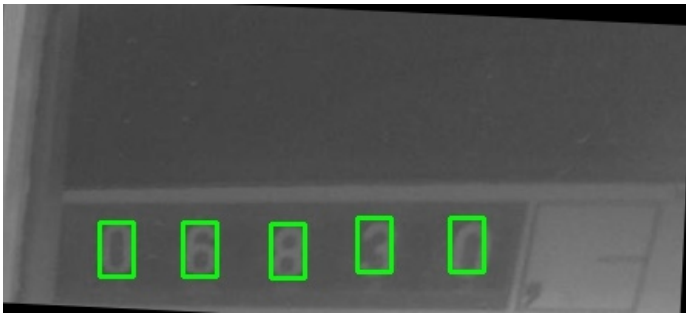


Figura 46: Dígits trobats a la imatge

Mirem quins són els valors que s' han guardat a la base de dades durant els 30 minuts [Figura 47].

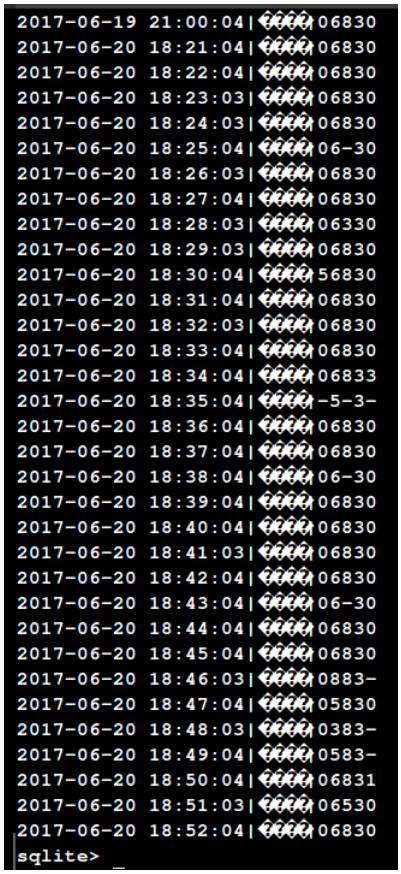


Figura 47: Contingut de la base de dades després del primer testeig amb el mirall

Com es pot apreciar, aquest cop també hi ha hagut alguns errors, i més que abans amb la càmara orientada directament al comptador, cosa que era d' esperar. Al final s' han près unes quantes mostres errònies seguides. Això pot també ser degut a un canvi significatiu en la lluminositat, o un lleuger canvi d' angle de la imatge presa.

Com ja s' ha dit, petits canvis poden alterar profundament el processat de la imatge, i per tant és convenient la màxima estabilitat del sistema durant les proves. Malgrat tot, la majoria de les mostres tenen el valor correcte.

Realitzem un altre testeig de 30 minuts, per tal de corroborar els resultats [Figura 48].

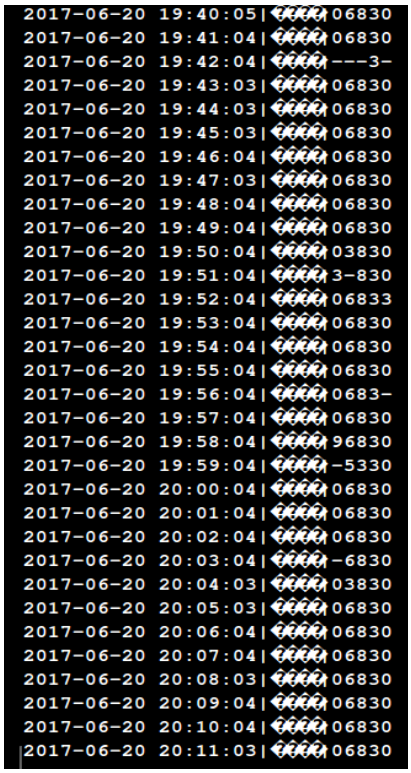


Figura 48: Contingut de la base de dades després del segon testeig amb el mirall

Veiem com el nombre d' errors és similar.

Amb això es pot deduïr que, tot i que el processat de la imatge a través del mirall és possible, la fiabilitat d' aquest disminueix significativament. Per tant, es requereix configurar el sistema de manera més precisa, i intentar evitar les inestabilitats tant del sistema com del seu entorn.

7.2. Testejos amb el comptador d' aigua

Les proves amb el comptador d' aigua són més complicades de fer que amb el comptador de llum, ja que en aquest cas s' han de fer tres processats diferents:

- El processat dels 4 dígits negres.
- El processat del dígit vermell.
- El processat de les agulles.

Per tant, s' ha de duur a terme més processat d' imatge i per tant el sistema és més sensible a errors.

Tenint això en compte, a continuació s' observarà els resultats de les proves que s' han duut a terme.

7.2.1. Càmara enfocant directament al comptador

En aquest testeig, s' ha mirat quin és el comportament del sistema quan es fan peticions de dades del comptador d' aigua, quan s' enfoca amb la càmara directament al comptador [Figura 49].

Figura 49: Montatge del sistema amb el comptador d' aigua

En aquest cas s' han realitzat testejos periòdics de menor duració, ja que com s' ha dit, el sistema ha de realitzar més processat i és més propens a fallades. Tot i així, com es demostrarà a continuació, el sistema és capaç d' enviar dades correctes.

El comptador d' aigua marca el valor 0026.1 per als dials. Per a les agulles que marquen els valors de les cèntèssimes, les milèssimes i les desenes de milèssima, el valor que dóna és més imprecís. Considerem que el sistema dóna un resultat correcte si ofereix valors entre 0026.19880 i 0026.19990 [Figura 50].

Figura 50: Imatge del comptador presa per la càmara

Per a executar les proves configurem la taula Cron [ref. 19] per tal que executi l' script `exec2.sh` cada minut, de manera que el Host envïi una petició de dades del comptador d' aigua a la Raspberry Pi [ref. 8]. Tal i com s' ha vist anteriorment, per la part de la Raspberry Pi, `pub.py` s' encarrega de la captura de la imatge, del seu processat i de l' enviament de dades cap al Host, i `sub.py` s' encarregarà de guardar les dades a la base de dades.

Com es pot observar [Figura 51], primerament es fa el processat dels dígits negres, a continuació es realitza la detecció del dígit vermell, i finalment es realitza el processat de les agulles, on es calcula l' angle d' inclinació i se n' extreu el valor del dial a partir d' aquest.

Un cop s ' obté tota la informació, aquesta s' ajunta i s' envia la dada, el nom de la imatge i la imatge.

```
pi@raspberrypi:~/Desktop/TFG/Scripts $ python3 pub.py
Connect result flag: 0
Connection succeded
Topic:  ocr_request
Request:  2: OCR image processing.
Detected lines angle:

0.0286622184761
-1.00357717599
-1.00307523898
Horizontal angle deviation: -0.6593300655

Number of contours found: 85

Number of filtered contours found: 12

Digits X positions not filtered: [207, 208, 139, 141, 147, 148, 246, 277, 72, 73, 77, 80]

Digits Y positions not filtered: [65, 66, 62, 63, 71, 73, 74, 61, 63, 64, 72, 74]

Most probable digit y position: 74

Number of digit contours found: 12 (should be 4)

Digits positions: [[72, 63], [73, 64], [77, 72], [80, 74], [139, 62], [141, 63], [147, 71], [148, 73], [207, 65], [208, 66], [24
6, 74], [277, 61]]

Digits positions (X not repeated, Y filtered): [[72, 63], [139, 62], [207, 65], [246, 74], [277, 61]]

Digits positions (filtered): [[72, 63], [139, 62], [207, 65], [277, 61]]

Digits interpreted from the image:

0
0
2
6
Detection of the red digit

Number of contours found: 72

Number of red filtered contours found: 2

24
64
Red digit value: 1
Total digits read from water counter: 00261
Detected lines angle:

-1.00357717599
0.0286622184761
Horizontal angle deviation: -0.487457478757

average angle for dial dial01: 108.247550014
average angle for dial dial001: 137.453337251
average angle for dial dial0001: 145.732427582
['94.0', '86.0', '84.0']
decimals values from needles: 9884
Water meter reading (Digit Counter plus Needles): 0026.19884
Image length: 239903
Image sent.
```

Figura 51: Flux d' execució de pub.py durant el processat del comptador d' aigua.

Aquest procés es repeteix periòdicament, fent que s' ompli la base de dades [Figura 52].

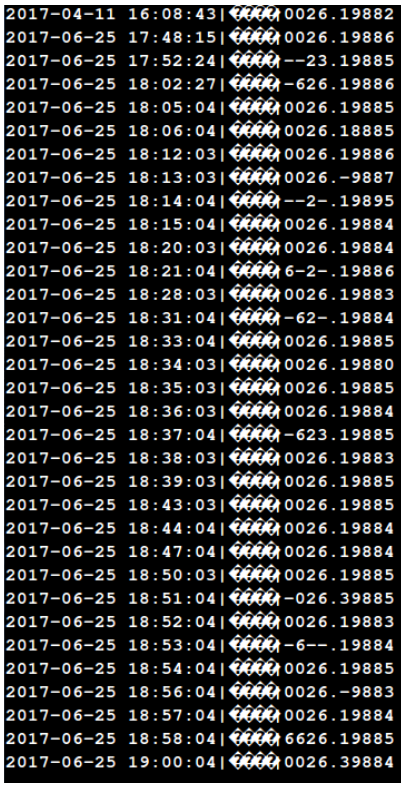


Figura 52: Contingut de la base de dades després del testeig amb el comptador d' aigua

Com es pot observar, hi ha alguns valors errònis. Recordem que el processat dels primers quatre dígits, del dígit vermell i de les agulles es fan per separat i després s' ajunta el valor total. Per tant, es dóna el cas que alguns cops falli una part del processat però les altres es realitzin correctament. Veiem però que el sistema és capaç d' obtenir informació correcta.

7.2.2. Càmara enfocant al mirall

En aquest testeig es realitzarà la lectura del comptador d' aigua amb la càmara enfocant al mirall [Figura 53]. En aquestes circumstàncies, la càmara no és capaç de capturar una imatge on es vegin les agulles del comptador correctament, i per tant només es podrà fer el processat dels dígits. Aquest és simplement un problema que es pot solucionar redissenyant la montura per tal que el mirall pugui reflectir tota la informació del comptador.

Figura 53: Montatge del sistema amb el comptador d' aigua, amb la càmara enfocant el mirall

Tot i així, aquest testeig servirà per comprovar que el sistema és capaç de detectar correctament els dígits i sobretot per demostrar que també pot detectar adequadament el dígit vermell, el qual és el més difícil de detectar i requereix un processat diferent.

Tal i com s' ha fet en els anteriors testejos, s' ha programat la taula Cron per tal que el Host faci peticions periòdiques cada minut a la Raspberry Pi [ref. 8], i aquest en faci el processat i n' enviï les dades.

En aquest cas torna a ser necessari rotar la imatge 180 graus i donar-li la volta horitzontalment. Un cop realitzat aquest canvi modificant els paràmetres de la funció `camera_setup`, i d' ajustar el codi per tal d' adaptar-se a la nova situació, es realitzen els testejos.

Observem com es pren la imatge [Figura 54], i el codi de processat alinia la imatge horitzontalment [Figura 55], filtra els contorns dels dígits [Figura 56], i en detecta la seva posició [Figura 57].

Figura 54: Captura de la reflexió del mirall del comptador d' aigua,voltejada i orientada correctament

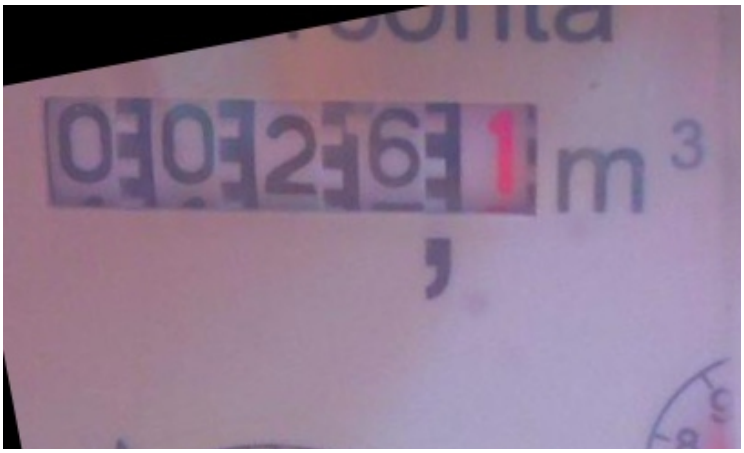


Figura 55: Imatge orientada hortizontalment

Figura 56: Dígits trobats a la imatge

Figura 57: Dígits trobats a la imatge

A continuació, es pot veure com s' aïlla el dígit de color vermell, facilitant-ne la seva detecció [Figura 58].

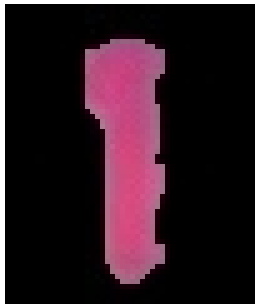


Figura 58: Dígit vermell després d' aplicar la màscara de color

Després del testeig, mirem els valors que s' han guardat a la base de dades [Figura 59].

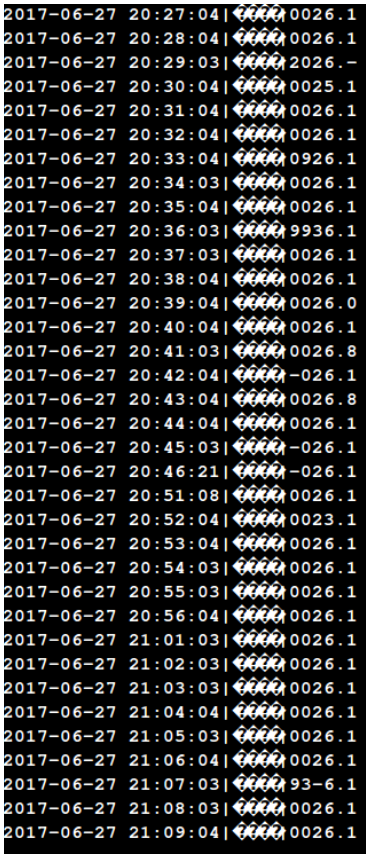


Figura 59: Contingut de la base de dades després de realitzar el testeig

Es veu doncs, que el sistema en general és capaç de processar generalment bé les imatges, tot i que com és habitual hi ha algunes mostres errònies. Cal recordar que el processat del dígit vermell es fa per separat, i per tant la possibilitat d' error és més alta. Tot i així, es demostra que es pot realitzar bé el processat del comptador d' aigua amb el mirall, i que es pot detectar bé el dígit vermell, el qual és el més delicat de processar.

7.3. Testejos de canvi de paràmetres de la càmara

Tal i com s' ha explicat en l' apartat **3.2.4.2.**, el sistema incorpora la possibilitat de cambiar els paràmetres de la càmara a través del protocol MQTT [ref 29]. S' han realitzat alguns testejos per comprovar que realment aquesta funcionalitat és utilitzable.

A continuació es mostra com intreactuen el Host i la Raspberry Pi [Figura 60][Figura 61].

```
pi@raspberrypi:~/Desktop/TFG/Scripts $ python3 pub.py
Connect result flag: 0
Connection succeded
Topic:  cam_request
Request:  HD,15,0,50,50,50,n,n: Camera settings modification.
['HD', '15', '0', '50', '50', '50', 'n', 'n']
Camera settings modified successfully.
```

Figura 60: Flux d' execució de pub.py durant la petició de canvi de paràmetres de la càmara.

```
user@TOSHIBA:~/Escritorio/QUART/TFG/rasp$ python sub.py 3
Connect result flag: 0
Connection succeded
Camera settings modification request.
Enter camera resolution ('widthheight' or 'HD'): HD
Enter camera framerate: 15
Enter camera rotation angle: 0
Enter camera sharpness: 50
Enter camera brightness: 50
Enter camera saturation: 50
Flip image vertically? ('yes'/'no'): n
Flip image horizontally? ('yes'/'no'): n
Request sent.
```

Figura 61: Flux d' execució de pub.py durant la petició de canvi de paràmetres de la càmara.

Després de realitzar algunes proves modificant els paràmetres, podem veure que, efectivament, els paràmetres de la càmara es modiquen correctament, i en les imatges capturades es poden apreciar les modificacions. [Figura 62][Figura 63][Figura 64].

Figura 63: Imatge amb els paràmetres de la càmara modificats (mostra 1)

Figura 62: Imatge amb els paràmetres de la càmara modificats (mostra 2)

Figura 64: Imatge amb els paràmetres de la càmara modificats (mostra 3)

8. CONCLUSIONS

Tal i com s' ha pogut observar durant els testejos, el sistema és capaç d' obtenir valors correctes d' ambdós comptadors, tant amb la càmara enfocant directament al comptador com fent ús dels miralls, però en ambdós casos es prenen un cert nombre de mostres errònies.

La causa d' això es deu en part a la imprecisió de les funcions que ofereixen els mòduls `cv2` i `pytesseract`. Aquestes funcions no ofereixen un processat fiable al 100%, i això és font d' errors. Per tant, a l' hora de desenvolupar el codi del processat s' ha de procurar combatre aquesta ineficiència fent un codi el més robust possible, aprenent d' aquests errors, i adaptar-s' hi.

Malgrat que el codi s' ha ideat per tal de minimitzar els errors, és probable que de cara a una versió més avançada que aquest prototipus, els errors de processat es puguin disminuir encara més a base de retocar i adaptar el codi a les necessitats del sistema. Tot i així és difícil assolir una precisió del 100%, sobretot amb el processat del comptador d' aigua, en el qual hi ha més elements a processar.

Malgrat tot, ha quedat demostrat que el processat dels comptadors és possible i que el sistema pot oferir valors de lectura correctes. També s' ha vist que es poden guardar aquests valors en una base de dades remota i visualitzar-ne el seu contingut amb altres eines com l' Excel o similars. Per tant, els resultats són positius.

La conclusió final és, per tant, que malgrat les imprecisions que les funcions de les llibreries de processat d' imatge utilitzades aporten al sistema, és possible adaptar-s' hi per tal de minimitzar-ne els errors, i és possible realitzar un sistema robust que és capaç de donar lectures correctes, i que a més, pugui evolucionar i millorar a mesura que es desenvolupa.

9. BIBLIOGRAFIA

[ref. 1] *IoT Based Data Processing for Automated Industrial Meter Reader using Raspberry Pi* [article]. Prachi H. Kulkarni , Pratik D. Kute , V. N. More. IEEE Explore. [Consultat: 20 de febrer de 2017]

[ref. 2] *Optical Character Recognition System for Seven Segment Display Images of Measuring Instruments* [article]. Rakhi P. Ghugardare, Sandip P. Narote, P. Mukherji, Prathamesh M. Kulkarni. IEEE Explore. [Consultat: 20 de febrer de 2017]

[ref. 3] *A Low cost Data Acquisition System From Digital Display Instruments Employing Image Processing Technique* [article]. S. Ghosh, S.Shit. IEEE Explore. [Consultat: 20 de febrer de 2017]

[ref. 4] *Recognition of Numerals on Digital Meters in Dynamic Measuring* [article]. C. Xing-chen, D. Hui-chuan, W. Jin-ling. IEEE Explore [Consultat: 20 de febrer de 2017]

[ref. 5] Pi NoIR Camera V2 [en línia]. Raspberry Pi. Adreça URL: https://www.raspberrypi.org/products/pi-noir-camera-v2/ [Consultat: 7 de febrer de 2017]

[ref. 6] Pi Camera V2 [en línia] Raspberry Pi. Adreça URL: https://www.raspberrypi.org/products/camera-module/ [Consultat: 7 de febrer de 2017]

[ref. 7] Pi Camera Python Module Manual [en línia]. Dave Jones. Adreça URL: https://picamera.readthedocs.io/en/release-1.13/ [Consultat: 14 de febrer de 2017]

[ref. 8] Raspberry Pi 3 Model B [en línia]. Raspberry Pi. Adreça URL: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/ [Consultat: 7 de febrer de 2017]

[ref. 9] Python [en línia]. Python. Adreça URL: https://www.python.org/ [Consultat: 7 de febrer de 2017]

[ref. 10] Python math module [en línia]. Python. Adreça URL: https://docs.python.org/2/library/math.html [Consultat: 26 de febrer de 2017]

[ref. 11] Python time module [en línia]. Python. Adreça URL: https://docs.python.org/2/library/time.html [Consultat: 26 de febrer de 2017]

[ref. 12] OpenCV cv2 Python module [en línia]. Python.Adreça URL: https://pypi.python.org/pypi/opencv-python [Consultat: 20 de febrer de 2017]

[ref. 13] Python PIL module [en línia]. Python. Adreça URL: http://www.pythonware.com/products/pil/ [Consultat: 22 de febrer de 2017]

[ref. 14] Python Pytesseract module [en línia]. Tesseract. Adreça URL: https://pypi.python.org/pypi/pytesseract/0.1 [Consultat: 7 de febrer de 2017]

[ref. 15] Python numpy module [en línia]. Python. Adreça URL: https://pypi.python.org/pypi/numpy [Consultat: 1 de març de 2017]

[ref. 16] OpenCV practice: OCR for the electricity meter [en línia]. M. Kompf. Adreça URL: https://www.mkompf.com/cplus/emeocv.html [Consultat: 1 de març de 2017]

[ref. 17] Python MQTT module paho-mqtt [en línia]. Mosquitto. Adreça URL: https://mosquitto.org/documentation/python/ [Consultat: 1 de març de 2017]

[ref. 18] Pyhton SQLite3 module [en línia]. Python. Adreça URL: https://docs.python.org/2/library/sqlite3.html [Consultat: 12 de març de 2017]

[ref. 19] Cron tables quick reference [en línia]. Admin's Choice. Adreça URL: http://www.adminschoice.com/crontab-quick-reference [Consultat: 15 d' abril de 2017]

[ref. 20] OpenSCAD [en línia]. OpenSCAD .Adreça URL: http://www.openscad.org/ [Consultat: 20 d' abril de 2017]

[ref. 21] SQLite [en línia]. SQLite. Adreça URL: https://www.sqlite.org/ [Consultat: 1 de març de 2017]

[ref. 22] Pi camera resolution [en línia]. Pi camera docs. Adreça URL: http://picamera.readthedocs.io/en/release1.10/api_camera.html#picamera.camera.PiCamera.resolution [Consultat: 20 de febrer de 2017]

[ref. 23] Pi camera framerate [en línia]. Pi camera docs. Adreça URL: http://picamera.readthedocs.io/en/release1.10/api_camera.html#picamera.camera.PiCamera.framerate [Consultat: 20 de febrer de 2017]

[ref. 24] Pi camera rotation [en línia]. Pi camera docs. Adreça URL: http://picamera.readthedocs.io/en/release1.10/api_camera.html#picamera.camera.PiCamera.rotation [Consultat: 20 de febrer de 2017]

[ref. 25] Pi camera sharpness[en línia]. Pi camera docs. Adreça URL: http://picamera.readthedocs.io/en/release1.10/api_camera.html#picamera.camera.PiCamera.sharpness [Consultat: 20 de febrer de 2017]

[ref. 26] Pi camera brightness[en línia]. Pi camera docs. Adreça URL: http://picamera.readthedocs.io/en/release1.10/api_camera.html#picamera.camera.PiCamera.brightness [Consultat: 20 de febrer de 2017]

[ref. 27] Pi camera saturation [en línia]. Pi camera docs. Adreça URL: http://picamera.readthedocs.io/en/release1.10/api_camera.html#picamera.camera.PiCamera.saturation [Consultat: 20 de febrer de 2017]

[ref. 28] SQLite byte stream type (BLOB) [en línia]. Python. Adreça URL: https://sqlite.org/datatype3.html [Consultat: 1 de març de 2017]

[ref. 29] MQTT protocol [en línia]. MQTT. Adreça URL: http://mqtt.org/ [Consultat: 1 de març de 2017]

[ref. 30] Canny edge detection [en línia]. OpenCV documentation. Adreça URL: http://docs.opencv.org/trunk/da/d22/tutorial_py_canny.html [Consultat: 8 de març de 2017]

[ref. 31] Hough line transform [en línia]. OpenCV documentation. Adreça URL: http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/hough_lines/hough_lines.html [Consultat: 9 de març de 2017]

[ref. 32] Contours [en línia]. OpenCV documentation. Adreça URL: http://docs.opencv.org/trunk/d4/d73/tutorial_py_contours_begin.html [Consultat: 13 de març de 2017]

[ref. 33] Optical Character Recognition [en línia]. Wikipedia. Adreça URL: https://en.wikipedia.org/wiki/Optical_character_recognition [Consultat: 1 de febrer de 2017]

[ref. 34] cv2.cvtColor [en línia]. OpenCV documentation. Adreça URL: http://docs.opencv.org/3.2.0/df/d9d/tutorial_py_colorspaces.html [Consultat: 8 d' abril de 2017]

[ref. 35] cv2.cvtCanny [en línia]. OpenCV documentation. Adreça URL: http://docs.opencv.org/trunk/dd/d1a/group__imgproc__feature.html#ga04723e007ed888ddf11d9ba04e2232de [Consultat: 1 de març de 2017]

[ref. 36] cv2.HougLines [en línia]. OpenCV documentation. Adreça URL: http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_houghlines/py_houghlines.html [Consultat: 5 de març de 2017]

[ref. 37] cv2.getRotationMatrix2D [en línia]. OpenCV documentation. Adreça URL: http://docs.opencv.org/trunk/da/d6e/tutorial_py_geometric_transformations.html [Consultat: 8 de març de 2017]

[ref. 38] cv2.warpAffine [en línia]. OpenCV documentation. Adreça URL: http://docs.opencv.org/trunk/d5/df1/group__imgproc__hal__functions.html#ga8a534cca6fb845c9ac77f10c35f67c0c [Consultat: 8 de març de 2017]

[ref. 39] cv2.findContours [en línia]. OpenCV documentation. Adreça URL: http://docs.opencv.org/trunk/d3/dc0/group__imgproc__shape.html#ga17ed9f5d79ae97bd4c7cf18403e1689 [Consultat: 10 de març de 2017]

[ref. 40] cv2.drawContours [en línia]. OpenCV documentation. Adreça URL: http://docs.opencv.org/trunk/d6/d6e/group__imgproc__draw.html#ga746c0625f1781f1ffc9056259103edbc [Consultat: 10 de març de 2017]

[ref. 41] cv2.boundingRect [en línia]. OpenCV documentation. Adreça URL: http://docs.opencv.org/trunk/d3/dc0/group__imgproc__shape.html#gacb413ddce8e48ff3ca61ed7cf626a366 [Consultat: 12 de març de 2017]

[ref. 42] Counter.most_common [en línia]. Python collections documentation. Adreça URL: https://docs.python.org/2/library/collections.html [Consultat: 12 de març de 2017]

[ref. 43] cv2.drawContours [en línia]. OpenCV documentation. Adreça URL: http://docs.opencv.org/3.1.0/d6/d6e/group__imgproc__draw.html#ga07d2f74cadcf8e305e810ce8eed13bc9 [Consultat: 12 de març de 2017]

[ref. 44] cv2.adaptiveThreshold [en línia]. OpenCV documentation. Adreça URL: http://docs.opencv.org/3.1.0/d7/d4d/tutorial_py_thresholding.html [Consultat: 12 de març de 2017]

[ref. 45] Python pytesseract module [en línia]. Python Pytesseract. Adreça URL: https://pypi.python.org/pypi/pytesseract/0.1 [Consultat: 7 de febrer de 2017]

[ref. 46] Pytesseract: recognition of single digits [en línia]. Github. Adreça URL: https://github.com/tesseract-ocr/tesseract/wiki/FAQ#how-do-i-recognize-only-digits [Consultat: 14 de març de 2017]

[ref. 47] HSL & HSV [en línia]. Wikipedia. Adreça URL: https://en.wikipedia.org/wiki/HSL_and_HSV [Consultat: 7 d' abril de 2017]

[ref. 48] Hough Circle Transform [en línia]. OpenCV documentation. Adreça URL: http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/hough_circle/hough_circle.html [Consultat: 1 d' abril de 2017]

[ref. 49] cv2.COLOR_BGR2HSV [en línia]. OpenCV documentation. Adreça URL: http://docs.opencv.org/3.1.0/d7/d1b/group__imgproc__misc.html#gga4e0972be5de079fed4e3a10e24ef5ef0aa4a7f0ecf2e94150699e48c79139ee12 [Consultat: 7 d' abril de 2017]

[ref. 50] cv2.inRange [en línia]. OpenCV documentation. Adreça URL: http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_colorspaces/py_colorspaces.html [Consultat: 8 d' abril de 2017]

[ref. 51] cv2.bitwise_and [en línia]. OpenCV documentation. Adreça URL: http://docs.opencv.org/2.4/modules/core/doc/operations_on_arrays.html [Consultat: 8 d' abril de 2017]

[ref. 52] cv2.HoughCircles [en línia]. OpenCV documentation. Adreça URL: http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_houghcircles/py_houghcircles.html [Consultat: 1 d' abril de 2017]

[ref. 53] Houghline Transfom Image [en línia]. OpenCV documentation. Adreça URL: http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_houghlines/py_houghlines.html [Consultat: 9 de març de 2017]

[ref. 54] cv2.line [en línia]. OpenCV documentation. Adreça URL: http://docs.opencv.org/3.1.0/d6/d6e/group__imgproc__draw.html#ga7078a9fae8c7e7d13d24dac2520ae4a2 [Consultat: 9 de març de 2017]

[ref. 55] np.arctan2 [en línia]. Numpy manual. Adreça URL: https://docs.scipy.org/doc/numpy/reference/generated/numpy.arctan2.html [Consulat: 1 d' abril de 2017]

[ref. 56] Mosquitto test server page [en línia]. Mosquitto. Adreça URL: https://test.mosquitto.org/ [Consultat: 7 de març de 2017]

[ref. 57] mqtt.client [en línia]. Python paho-mqtt. Adreça URL: https://pypi.python.org/pypi/paho-mqtt/1.1#constructor-reinitialise [Consultat: 2 de març de 2017]

[ref. 58] mqtt.connect [en línia]. Python paho-mqtt. Adreça URL: https://pypi.python.org/pypi/paho-mqtt/1.1#connect-reconnect-disconnect [Consultat: 2 de març de 2017]

[ref. 59] on_connect [en línia]. Python paho-mqtt. Adreça URL: https://pypi.python.org/pypi/paho-mqtt/1.1#callbacks [Consultat: 2 de març de 2017]

[ref. 60] on_publish [en línia]. Python paho-mqtt. Adreça URL: https://pypi.python.org/pypi/paho-mqtt/1.1#callbacks [Consultat: 2 de març de 2017]

[ref. 61] on_message [en línia]. Python paho-mqtt. Adreça URL: https://pypi.python.org/pypi/paho-mqtt/1.1#callbacks

[ref. 62] on_subscribe [en línia]. Python paho-mqtt. Adreça URL: https://pypi.python.org/pypi/paho-mqtt/1.1#callbacks [Consultat: 2 de març de 2017]

[ref. 63] csv Python module [en línia]. Python documentation. Adreça URL: https://docs.python.org/2/library/csv.html [Consultat: 20 d' abril de 2017]

[ref. 64] UNICODE UTF-8 [en línia]. Unicode character table. Adreça URL: https://unicode-table.com/en/

[ref. 65] Raspberry Pi Kit [en línia]. Amazon. Adreça URL: https://www.amazon.es/Raspberry-OficialStarterCargadordisipadores/dp/B01D0I5UXK/ref=sr_1_1s=electronics&ie=UTF8&qid=1496420956&sr=1-1&keywords=Raspberry+Pi+3+Official+Starter+Kit+White [Consultat: 7 de febrer de 2017]

[ref. 66] Pi Camera NoIR V2.1 [en línia]. Amazon. Adreça URL: https://www.amazon.es/dp/B01ER2SMHY/ref=pe_386191_124256311_TE_dp2 [Consultat: 7 de febrer de 2017]

[ref. 67] Leds [en línia]. Amazon. Adreça URL: https://www.amazon.es/SODIAL-Diodo-Emisor Ultrabrillante-Blanco/dp/B00CM5DEZO/ref=sr_1_2?ie=UTF8&qid=1496421237&sr=8 2&keywords=comprar+diodos+led [Consultat: 10 de maig de 2017]

[ref. 68] Conjunt de cargols i femelles [en línia]. Amazon. Adreça URL: https://www.amazon.com/Generic-Spacer-Assorted-Raspberry-Pi-Standoff/dp/B014J1ZLD6 [Consultat: 10 de maig de 2017]

[ref. 69] cv2.circle [en línia]. OpenCV documentation. Adreça URL: http://docs.opencv.org/3.1.0/d9/db7/group__datasets__gr.html#gga610754124ced68d1f05760b5948fbb76a6f0d8b2d9e3e947b2a5c1eff9e81ee95 [Consultat: 8 d' abril de 2017]

[ref. 70] cv2.HOUGH_GRADIENT [en línia]. OpenCV documentation. Adreça URL: http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_houghcircles/py_houghcircles.html [Consultat: 8 d' abril de 2017]

[ref. 71] Oil Level Reader [en línia]. C. Savina. Adreça URL: https://github.com/techsavi/OilLevelReader [Consultat: 8 de març de 2017]

[ref. 72] A-D converter the hard (but cheap!) way [en línia]. K. Sebesta. Adreça URL: http://www.eissq.com/DialADC.html [Consultat: 23 de març de 2017]

[ref. 73] Instrument Digitizer using Computer Vision [en línia]. E. Almeida i I. Vodopiviz. Adreça URL: https://hackaday.io/project/10617-instrument-digitizer-using-computer-vision [Consultat: 23 de març de 2017]

[ref. 74] Picamera [en línia]. Raspberry. Adreça URL: https://www.raspberrypi.org

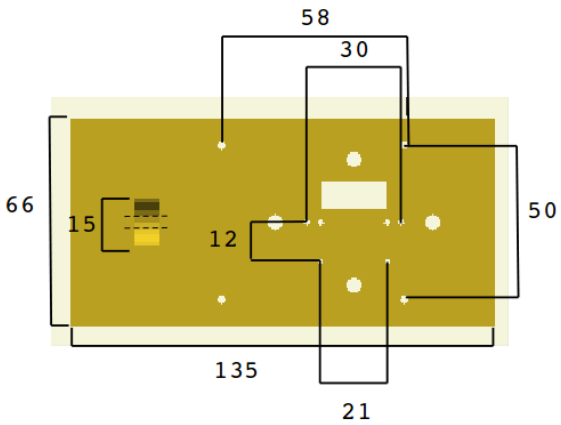
/documentation/usage/camera/python/README.md [Consultat: 23 de març de 2017]

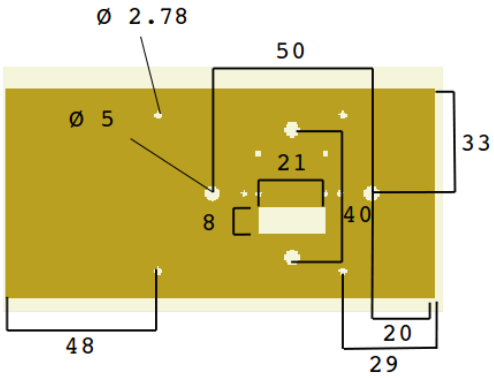
[ref. 75] Automatic Meter Reading AMR [en línia]. NYC Environmental Protection. Adreça URL: http://www.nyc.gov/html/dep/html/customer_services/amr_about.shtml [Consultat: 22 de juny de 2017]

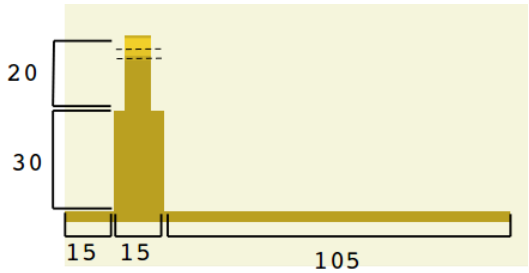
[ref. 76] Lentes delgadas [en línia]. Fisicalab. Adreça URL: [https://www.fisicalab.com/apartado/lentes-delgadas#contenidos [Consultat: 1 de juny de 2017]

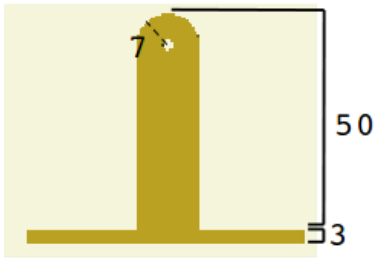
ANNEXOS

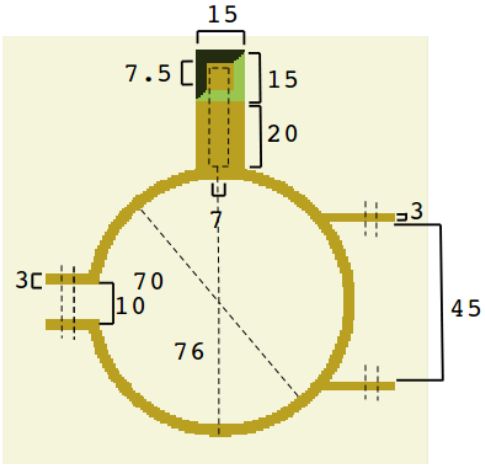
ANNEX I. Plànols de la montura

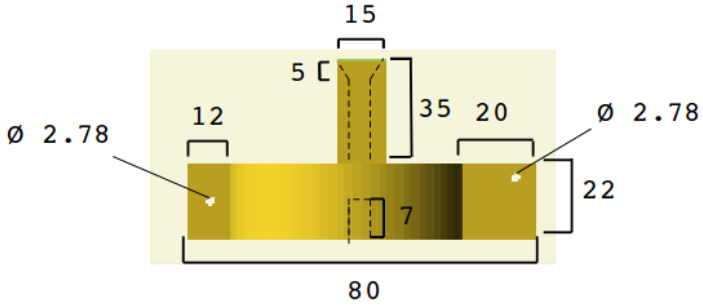


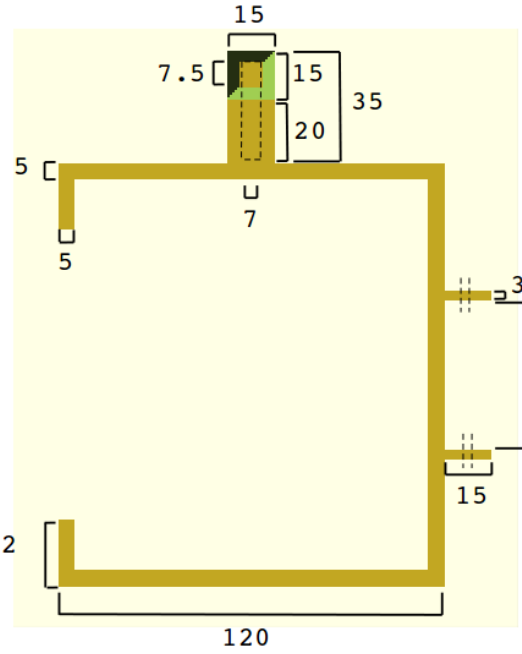


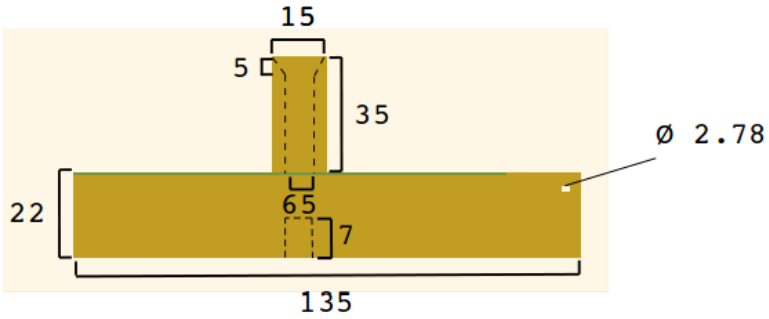


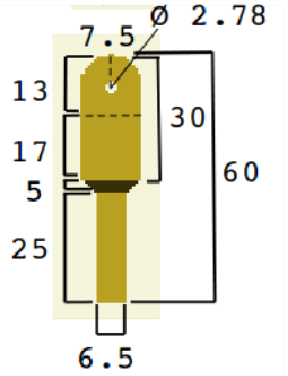

7
50
3

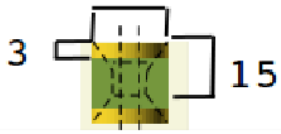

15
7.5
15
20
7
3
3
70
10
45
76

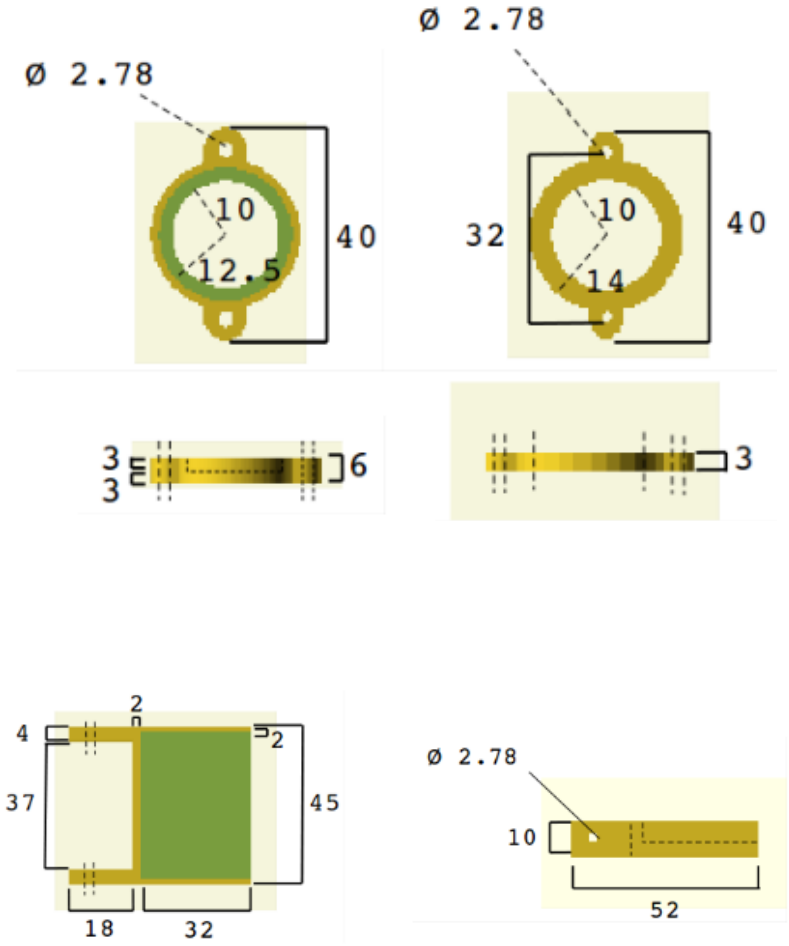

15
5
35
20
12
Ø 2.78
Ø 2.78
22
7
80


15
7.5
15
35
20
5
7
5
3
45
15
2
120


15
5
35
Ø 2.78
22
65
7
135


Ø 2.78
7.5
13
30
17
60
5
25
6.5


3
15


Ø 2.78
10
12.5
40
Ø 2.78
32
10
14
40
3
3
6
3
2
4
2
37
45
18
32
Ø 2.78
10
52